Lecture Notes in Computer Science 1873

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Mohamed Ibrahim Josef Küng
Norman Revell (Eds.)

Database and Expert Systems Applications

11th International Conference, DEXA 2000
London, UK, September 4-8, 2000
Proceedings

Springer

Volume Editors

Mohamed Ibrahim
Maritime Greenwich University Campus
Department of Computing and Mathematical Sciences
30 Park Row, London SE10 9LS, UK
E-mail: M.T.Ibrahim@gre.ac.uk

Josef Küng
University of Linz, FAW
Altenbergerstr. 69, 4040 Linz, Austria
E-mail: jkueng@faw.uni-linz.ac.at

Norman Revell
Middlesex University
Bounds Green, London, UK
E-mail: n.revell@mdx.ac.uk

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Database and expert systems applications : 11th international conference ; proceedings / DEXA 2000, London, UK, September 4 - 8, 2000. Mohamed Ibrahim . . . (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1873)
ISBN 3-540-67978-2

CR Subject Classification (1998): H.2, I.2.1, H.3, H.4, H.5

ISSN 0302-9743
ISBN 3-540-67978-2 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

Printed in Germany

Typesetting: Camera-ready by author
Printed on acid-free paper SPIN 10722345 06/3142 5 4 3 2 1 0

Preface

The Database and Expert Systems Applications (DEXA) conferences have established themselves as a platform for bringing together researchers and practitioners from various backgrounds and all regions of the world to exchange ideas, experiences and opinions in a friendly and stimulating environment. The papers presented at the conference represent recent developments in the field and important steps towards shaping the future of applied computer science and information systems.

DEXA covers a broad field: all aspects of databases, knowledge based systems, knowledge management, web-based systems, information systems, related technologies and their applications. Once again there were a good number of submissions: out of 183 papers that were submitted, the program committee selected 92 to be presented.

In the first year of this new millennium DEXA has come back to the United Kingdom, following events in Vienna, Berlin, Valencia, Prague, Athens, London, Zurich, Toulouse, Vienna and Florence. The past decade has seen several revolutionary developments, one of which was the explosion of Internet-related applications in the areas covered by DEXA, developments in which DEXA has played a role and in which DEXA will continue to play a role in its second decade, starting with this conference.

I would at this point like to express thanks to all the institutions which actively supported this conference and made it possible; namely:

- University of Greenwich
- Institut für Anwendungsorientierte Wissensverarbeitung (FAW), Universität Linz
- Austrian Computer Society
- DEXA Association

Many people contributed numerous hours to the success of this conference. Special thanks go to Maria Schweikert (Technische Universität Wien), Monika Neubauer and Gabriela Wagner (FAW, Universität Linz). We must also thank all members of the program committee, whose careful reviews are important to the quality of the conference, and the local organisers, who have put a lot of time and effort into making DEXA feel at home in Greenwich.

July 2000

Gerald Quirchmayr, Universität Wien, Austria
General Chair

Program Committee

General Chairperson:
G. Quirchmayr, University of Vienna, Austria

Conference Program Chairpersons:
M. Ibrahim, University of Greenwich, UK
J. Küng, FAW, University of Linz, Austria
N. Revell, Middlesex University, UK

Publication Chairperson:
V. Marik, Czech Technical University, Czech Republic

Workshop Chairpersons:
A M. Tjoa, Technical University of Vienna, Austria
R.R. Wagner, FAW, University of Linz, Austria
A. Al-Zobaidie, University of Greenwich, UK

Executive Chairperson
M. Everitt, University of Greenwich, UK

Local Arrangements
R.D. Finney, University of Greenwich, UK

Program Committee Members:
H. Afsarmanesh, University of Amsterdam, The Netherlands
A. Al-Zobaidie, University of Greenwich, United Kingdom
J. Albrecht, University of Erlangen-Nuremberg, Germany
B. Amann, CNAM&INRIA, France
F. Andres, NACSIS, Japan
K. Bauknecht, University of Zurich, Switzerland
T. Bench-Capon, University of Liverpool, UK
B. Bhargava, Purdue University, USA
J. Bing, NRCCL Oslo, Norway
P. Bosc, ENSSAT, France
O. Bukhres, Purdue University, USA
L. Camarinha-Matos, New University of Lisbon, Portugal
A. Cammelli, IDG-CNR, Italy
W.S. Cellary, University of Economics at Poznan, Poland
S. Chakravarthy, University of Florida, USA
S. Christodoulakis, University of Crete, Greece
P. Chrysanthis, University of Pittsburgh, USA
P. Ciaccia, University of Bologna, Italy
C. Collet, LSR-IMAG, France
C. Combi, University of Udine, Italy
D. Cowell, University of Greenwich, United Kingdom
W.B. Croft, University of Massachusetts, USA
J. Debenham, University of Technology, Sydney, Australia
M. Deen, University of Keele, UK
J. Eder, University of Klagenfurt, Austria
T. Eiter, Technical University of Vienna, Austria
G. Engels, University of Paderborn, Germany

P. Fankhauser, GMD, Germany
E. Fernandez, Florida Atlantic University, USA
S. Field, IBM Zurich, Switzerland
P. Funk, Mälardalen University, Sweden
A.L. Furtado, University of Rio de Janeiro, Brazil
G. Gardarin, INRIA, France
S. Ghandeharizadeh, University of Southern California, USA
P. Godfrey, York University, Canada
G. Gottlob, Vienna University of Technology, Austria
P. Grefen, University of Twente, The Netherlands
U. Hahn, University of Freiburg, Germany
J.-L. Hainault, University of Namur, Belgium
A. Hameurlain, Université Paul Sabatier, France
I. Hawryszkiewycz, University of Technology, Sydney, Australia
Y. Kambayashi, University of Kyoto, Japan
M. Kamel, Naval Postgraduate School, USA
N. Kamel, American University in Cairo, Egypt
G. Kappel, University of Linz, Austria
D. Karagiannis, University of Vienna, Austria
K. Karlapalem, University of Science and Technology, Hong Kong
R. Keller, University of Montreal, Canada
M.H. Kim, KAIST, Korea
B. Knight, University of Greenwich, UK
J. Kouloumdjian, LISI-INSA, France
P. Kroha, Technical University of Chemnitz, Germany
J. Lazansky, Czech Technical University, Czech Republic
M. Leonard, University of Geneve, Switzerland
T.W. Ling, University of Singapore, Singapore
W. Litwin, University Paris 9, France
M. Liu, University of Regina, Canada
F. Lochovsky, University of Science and Technology, Hong Kong
P. Loucopoulos, University of Science and Technology in Manchester, United Kingdom
S.K. Madria, Purdue University, USA
A. Makinouchi, Kyushu University, Japan
H. Mannila, Microsoft Research, USA
V. Marik, Czech Technical University, Czech Republic
S. Marinai, University of Florence, Italy
S. Mazumdar, New Mexico Institute of Mining and Technology, USA
R. Meersman, Free University of Brussels, Belgium
E. Metais, University of Versailles, France
M. Mohania, Western Michigan University, USA
S. Monties, EPFL, Switzerland
T. Morzy, Poznan University of Technology, Poland
G. Müller, University of Freiburg, Germany
W. Naqvi, Raytheon Systems Company, USA
E. Neuhold, GMD-IPSI, Germany
G. Ozsoyoglu, University Case Western Research, USA
G. Pangalos, University of Thessaloniki, Greece
B. Pernici, Politecnico di Milano, Italy
G. Pernul, University of Essen, Germany
G. Quirchmayr, University of Vienna, Austria
F. Rabitti, CNUCE-CNR, Italy

I. Ramos, Technical University of Valencia, Spain
H. Reiterer, University of Konstanz, Germany
N. Revell, Middlesex University, UK
C. Rolland, University Paris I, France
E. Rundensteiner, Worcester Polytechnic Institute, USA
M.E. Rusinkiewicz, MCC, USA
D. Sacca, University of Calabria, Italy
M. Savonnet, University of Bourgogne, France
E. Schweighofer, University of Vienna, Austria
T. Sellis, NTU Athens, Greece
M. Smith, University of California, USA
G. Soda, University of Florence, Italy
H. Sonnberger, EUROSTAT, Luxemburg
U. Srinivasan, CIRO, Australia
O. Stepankova, Czech Technical University, Czech Republic
D. Stott Parker, University of Los Angeles, USA
V.S. Subrahmanian, University of Maryland, USA
J. Suzuki, Keio University, Japan
M. Takizawa, Tokyo Denki University, Japan
K. Tanaka, Kobe University, Japan
Z. Tari, RMIT, Australia
S. Teufel, University of Oldenburg, Germany
J. Teuhola, University of Turku, Finland
B. Thalheim, Technical University of Cottbus, Germany
J.M. Thevenin, University Paul Sabatier, France
C.H. Thoma, IT pro, Switzerland
R. Traunmüller, University of Linz, Austria
P. Triantafillou, Technical University of Crete, Greece
A. Tsalgatidou, University of Athens, University of Jyväskylä, Greece, Finland
S. Urban, Arizona State University, USA
K. Vidyasankar, Memorial University of Newfoundland, Canada
Y. Wand, University of British Columbia, Canada
M. Wing, Middlesex University, UK
W. Winiwarter, University of Vienna, Austria
A. Zaslawsky, Monash University, Australia

External Referees

I. Pramudiono, University of Tokyo, Japan
I. Karali, University of Athens, Greece
A. Pikrakis, University of Athens, Greece
C. Vasilakis, University of Athens, Greece
M. Nikolaidou, University of Athens, Greece
E. Berki, University of Jyvaskyla, Finland
J.H. Canós, Universitat Politecnica de Valencia, Spain
N. Rodriguez Brisaboa, Universitat Politecnica de Valencia, Spain
J.C. Casamayor, Universitat Politecnica de Valencia, Spain
E. Mena, Universitat Politecnica de Valencia, Spain
E. I. Benitez-Guerrero, IMAG-LSR Laboratory, France
G. Vargas-Solar, IMAG-LSR Laboratory, France
A. Lefebvre, France Telecom R&D, France

Table of Contents

Object Oriented, Object Relational II

Multimedia Databases II

Fundamentals for Applications II

Workflow

Security Aspects

Fundamentals for Applications III

XML

Advanced Databases I

Query Aspects I

Knowledge Aspects I

Data Warehouses I

Advanced Databases II

Query Aspects II

Knowledge Aspects II

Data Warehouses II

Advanced Databases III

Query Aspects III

Analysis and Design

Data Mining, Knowledge Discovery

Advanced Databases IV

Query Aspects IV

Using the Web I

Indexing

Applications I

Using the Web II

Distributed Aspects

Applications II

Multimedia Databases I - continuation

A Foolish Consistency: Technical Challenges in Consistency Management

Anthony Finkelstein

University College London, Department of Computer Science, Gower St., London WC1E 6BT UK
a.finkelstein@cs.ucl.ac.uk

Abstract. This paper outlines the area of consistency management and argues for its importance. A motivating example is presented to support the argument. The paper sets out the key technical challenges for research in this area. A broad research agenda is outlined with some signposting of particularly interesting directions.

1 What is Consistency Management?

"A foolish consistency is the hobgoblin of little minds adored by little statesmen and philosophers and divines. With consistency a great soul has simply nothing to do … speak what you think today in words as hard as cannonballs and tomorrow speak what tomorrow thinks in hard words again though it contradict everything you said today"

R.W. Emerson

The everyday concept of inconsistency is easy to grasp. It is simply saying something in one place and another contradictory thing in another place. The equivalent of the standard logical notion of inconsistency in which one asserts both that *the sky is blue* and that *the sky is not blue*.

Of course inconsistency in this form is not inherently a problem until you try and base some actions in the real world upon the inconsistent assertions - selecting a coat for example. The results of doing so are varied but, depending on the veracity of the assertions, the resulting actions will interfere – simultaneously requiring wearing a raincoat and not wearing a raincoat.

Inconsistency has many causes. Foremost among these is that it results from collaboration of multiple actors, each with different opinions, views and interpretations on the real world. It is also the result of uncertainty leading to the preservation of an equivocal and hence inconsistent position. Inconsistency commonly results from errors or more rarely from deliberate falsification.

It should be clear from this that there are many circumstances in which inconsistency is acceptable, indeed desireable. In general this is in any situation where you are seeking to establish what to do prior to committing to a course of action. In such cases you may wish to have all the alternatives laid before you and as full and accurate information about the state of the world as it is possible to obtain. The consequences

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 1–5, 2000.

of this are obvious. Rather than thinking about removing inconsistency we need to think about "managing consistency". This means preserving inconsistency where it is desirable to do so, identifying inconsistency at the point where decisions are required and removing (or otherwise remedying) inconsistency prior to taking action. This requires a major change in the way we think.

Classically our concern has been to organise our information management practices so as to prevent inconsistency from arising, or using database mechanisms to remove inconsistency as close to the source as possible. Where inconsistency occurs despite our best attempts to eliminate it is viewed as an "application level" concern to be handled on a case by case basis. In a closed and relatively static corporate environment this approach might be acceptable. In the "new" information management setting of federated organisations, web publication and distributed collaborative work it patently does not work. We would like flexible and open consistency management with strategies, policies and tools defined at the generic level.

2 A Motivating Example

To make this discussion less abstract let us consider an example - distributed software development. Software developers, physically distributed, engage in highly collaborative work. Opinions differ and multiple inconsistent requirements and design alternatives are a particular feature of the work. Deferring commitment is an important part of the developer's armoury. The products of software development work are expressed in a variety of complex formal and semi-formal languages. These formal and semi-formal languages have more or less precise interpretations in the domain of computation.

Potential interference in this domain is reflected as a consistency relation at the level of the language itself. A typical example (for UML notations) is that an instance in a collaboration diagram for a system must be an instance of a class that appears in the class diagram for that system. Such relations can be in the form of direct mappings between the languages or by reference to a shared meta-model.

Much of the information that software developers handle is semi-structured text. The possibilities for treating consistency in this context are limited. Either we can handle consistency at the level of the structure, a formal artefact, specifying for example that the interface functional specification must contain sections with the same headings as the user manual, or at the natural language level. If at the natural language level the best we can expect are heuristic strategies.

3 Why is Consistency Management Difficult?

In a trivial case such as *the sky is blue*, *the sky is not blue* the inconsistency is easy to spot, if not to deal with. By contrast inconsistencies in real settings such as software development present much more challenging problems.

- The assertions can be vague or semantically ungrounded so that it is difficult to determine their precise meaning. It may be impossible to determine how the asser-

tions relate to the real world and hence establish whether actions based on them will interfere. Assertions may be made in different (formal) languages whose relationship to each other is uncertain

- Inconsistencies can be hidden in a mass of other assertions. These assertions can be physically distributed in such a way that it is difficult to assemble the information so as to detect the inconsistency.
- An inconsistency can itself be distributed across many assertions. Consistency checking cannot always be reduced to pair-wise comparison. There may be many inconsistencies, some related and some independent, in a set of assertions. Each inconsistency can be of varying importance in the domain.
- The presence of inconsistency may pollute the set of assertions and hence prevent the use of certain technical mechanisms (such as standard first-order logic) for reasoning about the information.
- The information, which itself may be changing rapidly, may be intertwined with a complex workflow so that it is not easy to determine the points at which it is critical to establish consistency.
- In many cases inconsistencies reflect slips and minor errors or possibly delayed commitments which are relatively easy to resolve. Some inconsistencies however reflect serious conflicts with substantial knock-on consequences and may involve substantial negotiation.

4 What We Would Like To Do

Ideally what we would like to build a toolset which should include:

- tools to help establish, express and reason about the relationships between formal languages;
- tools to check consistency with respect to these relationships and to provide diagnostic feedback;
- where inconsistencies have been detected, tools to visualise the inconsistencies;
- tools to track inconsistencies and preserve diagnostic information in the face of ongoing change;
- tools to support resolution either through the removal of inconsistency – rectification – or by reducing the scope or severity of the inconsistency – amelioration;
- tools to support the rationale associated with consistency management.

In addition we need to be able to:

- specify policies with respect to when consistency should be checked and when resolution mechanisms should be applied;
- enforce these policies at appropriate times and in an appropriate manner.

5 Technical Challenges

To build such a toolset means surmounting technical challenges in a number of core areas of applied computer science.

We need an in-depth understanding of the semantics of formal languages used in particular application domains; software engineering is a good example of this. We need to be able to construct meta-models with a sound foundation and be able to reason across different formal languages expressed in terms of those meta-models. We should also be able to express constraints on the meta-model as consistency checks. In the case of an approach based on direct mappings we need "patterns" (for want of a better term) for expressing such mappings that mean the relationships are not constructed in an ad-hoc manner. More speculatively, where informal text is used we need to explore the possibility of using techniques from natural language processing to indicate potential inconsistencies.

We need to be able to check consistency in a way that is fast and highly scaleable. We can anticipate that we may have to check the consistency of very large numbers of highly complex distributed documents and other information artefacts. Consistency checking needs to be unobtrusive and low cost. This may mean devising algorithms that are computationally efficient in terms of space and performance. It may also mean devising strategies for distributing the consistency checking task and allowing the "user" to scope the consistency checks in some suitable manner. The consistency checking must be done in a way that yields useful diagnostic feedback at least localisation.

We need to devise visualisation techniques appropriate to showing consistency across complex information 'spaces'. Ideally the visualisation should also support consistency-based navigation through the information space in which it is possible to navigate across pieces of information linked by consistency relationships.

The visualisation techniques need to be linked to version and configuration control strategies that allow inconsistencies to be tracked as surrounding information changes. This is critical as we anticipate that known inconsistencies, particularly the controversial, hard to resolve ones, may be preserved as outstanding issues over relatively lengthy periods.

Conflict resolution is critical to consistency management and is both extremely difficult and little worked on, at least directly. Contributions from the fields of artificial intelligence particularly planning, computer-supported cooperative work and of course the social sciences are suggestive and provide some valuable concepts, but have yielded little which can be directly applied. Ammelioration – reducing the space, or perhaps the severity, of a conflict is an important element of resolution but demands that we have some sort of metric. This is problematic in many of the settings with which we are concerned.

It is reasonable to anticipate that users will wish to associate meta-data with inconsistencies. We have already discussed two classes of such meta-data – diagnostic and configuration information. Rationale indicating the history of discussion with respect to an inconsistency or set of inconsistencies is particularly important. Obviously this will need to be associated with the relevant information and preserved beyond the resolution of the inconsistency. The development of a rationale scheme suitable for this purpose constitutes an interesting challenge.

The specification and execution of consistency management policies provides a further technical challenge. A policy should set down what checks to perform at what

point in the workflow or process to which that information relates. It may further specify what actions should be taken by way of resolution, if any, in order that the workflow or process should continue. Clearly policies themselves are domain specific but the expression of policies is a general issue. The expression of a consistency management policy is however tied to the particular way in which the workflow is described and monitored, a highly complex issue in itself.

There is not scope in this paper to present a review of related work. Probably the most recent account, though biased towards software development is [4]. The Proceedings of the Viewpoints '96 Workshop [5] and the Multi-dimensional Separation of Concerns Workshop '00 [6] reflect the growing interest among software engineers in this issue. The author has a long sequence of work addressing consistency management that includes [1], [2], [3]. Each of these papers contains a discussion of relevant contributions, these include work in AI, CSCW and distributed heterogeneous databases.

6 Where Now?

The end goal is the construction of a lightweight and generic set of consistency management tools which can operate across heterogeneous and distributed information, to be offered as components and web services from which an information manager will be able to assemble a consistency management solution. This goal is some way off and, as has been shown above, there are some major hurdles to surmount. Having said this, the advent of XML (eXtensible Markup Language) and associated web technologies including XML-aware databases and query languages form a very powerful substrate on which to build the sort of consistency management we envisage. The contributions of the database community to the achievement of this are eagerly sought.

References

1. Easterbrook, S., Finkelstein, A., Kramer, J. & Nuseibeh, B.: Coordinating Distributed ViewPoints: The Anatomy of a Consistency Check, International Journal on Concurrent Engineering: Research & Applications, 2,3, (1994)
2. Emmerich, W., Finkelstein, A., Montangero, C., Antonelli, S., Armitage, S. & Stevens, R.: Managing Standards Compliance, IEEE Transactions on Software Engineering, 25, 6 (1999)
3. Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., & Nuseibeh, B.: Inconsistency Handling In Multi-Perspective Specifications, IEEE Transactions on Software Engineering, 20, 8, (1994),
4. Nuseibeh, B, Easterbrook, S. & Russo, A.: Leveraging Inconsistency in Software Development, IEEE Computer, 33, 4 (2000).
5. Viewpoints 96: An International Workshop on Multiple Perspectives in Software Development; ACM Symposium on Foundations of Software Engineering 1996 http://www.soi.city.ac.uk/~gespan/vptoc.html
6. Workshop on Multi-Dimensional Separation of Concerns in Software Engineering 2000, 22nd ICSE, http://www.research.ibm.com/hyperspace/workshops/icse2000/

The BORD Benchmark for Object-Relational Databases†

Sang Ho Lee[1], Sung Jin Kim[1], and Won Kim[2*]

[1] School of Computing, Soongsil University, Seoul, Korea
{shlee, sjkim}@computing.soongsil.ac.kr
[2] Cyber Database Solutions, Austin, Texas, U.S.A.
won.kim@cyberdb.com

Abstract. This paper describes a new benchmark for object-relational DBMSs, the Benchmark for Object-Relational Databases (BORD). BORD has been developed to evaluate system performance peculiar to the object-relational data model as well as to the functions of modern database systems. The design philosophy, benchmark database, and test queries of the BORD benchmark are presented. BORD features scaleability, use of a synthesized database only, and a query-oriented evaluation. In total, thirty-six test queries have been designed under fifteen categories, which include exact matches, object identifiers, joins, class references, set-valued attributes, user-defined methods, built-in versus user-defined operators, flattening queries, generic abstract data types (ADTs), and spatial ADTs. In order to show the feasibility of the benchmark, we have implemented BORD with two commercial object-relational database systems. The experimental results are also reported.

1 Introduction

Since UniSQL, Inc. ushered in the era of object-relational database (ORDB) technology in 1992, ORDBs have been regarded as the next-generation database system that will dominate the database field. An ORDB incorporates an object-oriented data model in a way that it is compatible with a relational database (RDB). This compatibility allows us to preserve all benefits of and investment in relational database technology. Vendors have already started to ship new database systems labeled "object-relational" into the marketplace.

A number of database benchmarks are proposed in the literature [9, 6]. They include the TP1/DebitCredit benchmark, the Transaction Processing Council (TPC) series [6], the Wisconsin benchmark [6], the AS^{3}AP benchmark [6], the Set Query benchmark [6], the HyperModel [1], the OO1 benchmark [5], the OO7 benchmark [3, 4], the Sequoia 2000 storage benchmark [11], the Justitia benchmark [10], the BUCKY benchmark [2], and so on. To our best knowledge, only the BUCKY

† This work is supported in part by the University Foundation Research (supervised by IITA) of the Ministry of Information & Communication of Korea.
* This author's research was partially funded by the Brain Korea-21 Research Program.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 6–20, 2000.

benchmark is a query-oriented benchmark for ORDBs. For detailed descriptions of various benchmarks, see [6, 9].

In this paper, we describe a new benchmark for DBMSs, the Benchmark for Object-Relational Databases (BORD). BORD is a generic benchmark for ORDBs. It measures performance of DBMS functions that should be provided by typical ORDBs or by modern database systems. In particular, the useful features that an object-relational data model should provide from the point of the end user, have been incorporated in the design phase of BORD. BORD uses a synthesized database only, which both helps portability of BORD and allows us to verify the correctness of test queries easily. BORD is scaleable so that it can run on a small computer and a large-scale computer by adjusting the volume of the test database accordingly.

This paper is organized as follows: The design philosophy is presented in section 2, and the schema and instances of the test database are described in section 3. In section 4, test queries for ORDBs are given. Section 5 describes our implementation experience of BORD with two commercial ORDBs, and also shows our experimental performance results. Section 6 contains closing remarks.

2 Design Philosophy

BORD is designed to be a generic benchmark, which should be useful for a large group of potential customers with relatively standard applications. It is not intended to be domain-specific. Rather, the target of BORD is a general-purpose ORDB that can be used in a wide variety of application domains.

In order to design a benchmark for an ORDB, we need to define precisely what an object-relational data model is as well as what an ORDB should provide to the user in terms of DBMS functions. Generally speaking, an ORDB should subsume a RDB in term of functions, and should incorporate beneficiary functions (for example, encapsulation, inheritance, etc.) of an OODB in the context of SQL. Unlike a relational data model, an object-relational data model does not have a reference document that is widely accepted as standard. Even though many database vendors have started to release their products labeled as ORDBs, we have found that there is considerable function discrepancy in these ORDBs. Stonebraker [12] identified the four main characteristics of an ORDB, which should be supported by systems. Kim [8] proposed a practical metric for object-relational completeness, which may serve as a reference for determining whether a product is an ORDB, and for determining the degree of completeness of the product. He grouped ORDB capabilities into two categories (i.e. primary and secondary) that list a number of features for DBMSs to be qualified as true ORDBs. Our perspective on ORDBs is mainly based on [8] and [12]. In addition, we have looked at several commercial ORDBs available on the market in terms of their functions. Principle features of current ORDBs have been considered, and have been incorporated into the object-relational data model that is used in this paper.

BORD is designed to benchmark an ORDB only. It is not intended for RDBs or OODBs, and consequently is not meant to be a substitute for well-known RDB or OODB benchmarks. Little consideration of RDB-specific and OODB-specific

features has been given. For example, the pointer swizzling of OODBs is excluded in BORD since it is a primary operation of OODBs. Similarly, aggregate functions (such as count, average, sum, minimum, and maximum) of RDBs have been excluded. The functions of ORDBs at which BORD aims are part of the functions that differentiate ORDBs from RDBs distinctively. These functions correspond to an extension of a relational data model to accommodate the good features of an object-oriented data model.

Commercial ORDBs differ widely in terms of the functions they offer the user. For example, some systems allow an index to be created for spatial data types, while others do not. Even though missing functions could be implemented in a database application program, and therefore be manually simulated by database programmers, BORD does not permit such a hand-coded module in an application program. Our position is that a performance evaluation should be carried out against a target product as it is, so an add-on module is prohibited in our benchmark. The underlying idea behind this is that we would like to give considerable credit to database systems that offer good features to users, as long as the features are useful to users and are indispensable to ORDBs.

One important consideration in the design phase of BORD is scaleability. The volume of a database has an intrinsic performance impact on DBMS, and so should be chosen carefully. The volume of databases in database applications is getting larger and larger as computer technology advances, and the popularity of multimedia information accelerates the expansion of database volume greatly. The recent technical advance in memory devices is also fascinating. BORD approaches this matter in a way that allows a user to specify the volume of the test database by use of a scale factor. A nominal database size is given, and the user is expected to scale the size of the test database accordingly, to reflect a particular database application.

BORD uses synthetically generated relations only, instead of empirical data from real databases. One evident problem with empirical databases is that they are difficult or impossible to scale. Empirical databases are not likely to be flexible enough to permit the systematic benchmark of databases systems. For example, with empirical databases it is very difficult to have a selection query with fixed (say, 10%) selectivity. Empirical databases do have one advantage in that they give the user a more realistic feeling than synthesized databases do. However, an empirical database, which is just a snapshot of a real database application, is never representative of all generic database applications. On the other hand, in the case of synthesized databases, we are able to have full control of database generation. Hence, we are able to generate test databases as we intend to do. Synthesized databases are indeed scaleable if they are well designed.

The query optimizers of DBMSs usually speed up the execution of queries by finding the best execution plan among many plausible ones. Almost all efficient execution plans utilize the presence of indices. While some benchmarks (such as the Wisconsin benchmark, the Set Query benchmark) put stringent restrictions on the creation of indices in the course of DBMS evaluation, BORD allows indices to be created as needed. In principle, any potentially useful indices may be created as long as the DBMS supports them. The data types of the attributes on which indices are defined may vary significantly, depending on the functions of ORDBs. Indices in BORD may be created not only on conventional data types (such as integer, numeric,

and string) but also on sets, methods, OIDs, etc. As for indices themselves, they are not confined to conventional B^+-trees in our benchmark. Indices in BORD may be one of the B-tree family, one of the R-tree family, one of various hashing schemes, or any useful schemes, as long as the ORDB under consideration supports them and the indices seem to help to evaluate test queries. The idea behind this is that DBMSs that offer useful features should receive more credit than those that do not.

We believe that a single benchmark metric is not sufficient to show the performance characteristics of ORDBs. A number of metrics are included in our benchmark results. BORD has a number of test queries (see section 4). For each BORD query, an average elapsed time, which is an arithmetic average of ten cold runs, is reported. The cold run occurs if the database buffer of the DBMS is empty when queries are submitted for execution. BORD also computes queries per minute (QPM), that is, the number of successfully executed queries per minute. It also reports "Price per QPM (P/QPM)" as the final metric of the DBMS, in which the price, as in the TPC D benchmark, denotes all incurring costs for hardware and software as well as five-year maintenance costs. The inclusion of the price factor as the final metric helps the scaleability of BORD.

3 Schema and Instances of Test Databases

The schema and instances of BORD are constructed after a typical school database application. Fig. 1 shows the schema of the test database. There is no multiple inheritance in the BORD schema. Note that there are 17 classes that constitute a seven-depth inheritance hierarchy, and a four-hop class composition hierarchy

It should be noted that each class has the *name* attribute. The *name* attribute is not used in test queries at all, but is used to shuffle the instances of classes completely before test databases are loaded into an ORDB. The *name* attribute will be further explained when implementation issues are described.

The *city* attribute and the *state* attribute in the *Person* class (including its subclasses, of course) have a 1% and 10% key average distribution, respectively. The *city* attribute of the subclasses of the *Person* class (i.e., *Student, Graduate, Professor, External, Fulltime, Parttime, TA,* and *RA*) always has one of a hundred different values (i.e., 'city00', 'city01', …, 'city99'), which are uniformly distributed. Similarly, the *state* attribute has one of ten different values (i.e., 'S0', 'S1', …, 'S9'), which are also uniformly populated. The values of the *age* attribute are randomly distributed and range from 20 to 60. Moreover, they are populated so that persons with age sixty constitute exactly 1% of all people. The values of the *gender* attribute of the *Person* class are evenly distributed, so that the *gender* attribute has a 50% key average distribution. The *rate* attribute of the *Employee* class ranges between 0.00 and 0.99.

The *Employee* class has a method, *salary*, which is inherited by the subclasses. The *TA* class has its own method, *salary,* with the same name. The *major* attribute of the *Student* class is a foreign key that references the *departmentId* attribute of the *Department* class. In our schema, there are four generic abstract data types (ADTs),

rank_t, time_t, c_project_t, and *p_project_t*. Those ADTs will be explained in section 4.

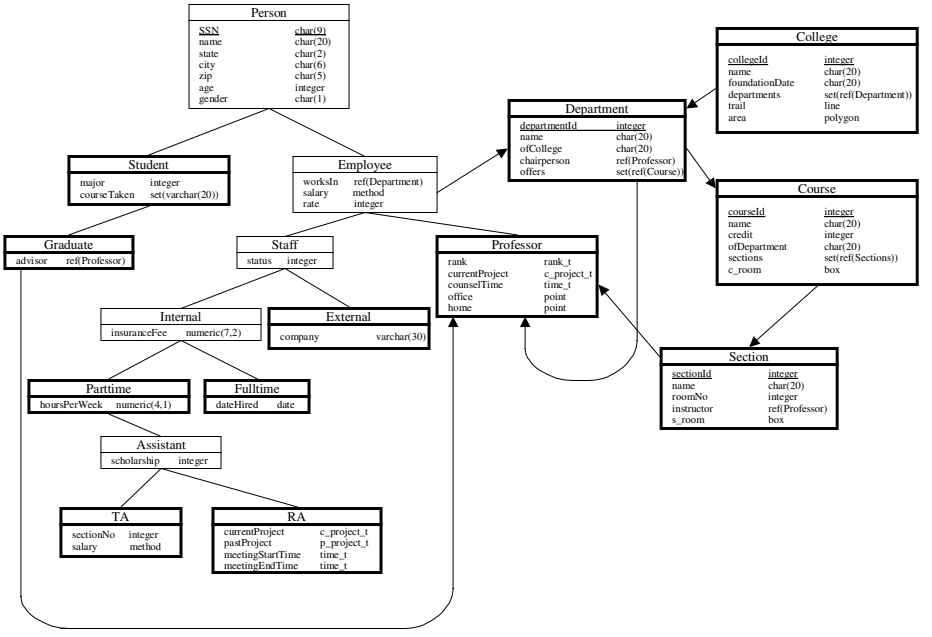


Fig. 1. Test database schema. The regular rectangle denotes a class without instances, and the thick rectangle represents a class with instances. The regular line denotes an inheritance relationship. The line with an arrow denotes a class reference relationship. The underlined attribute in each class represents a key attribute

The *courseTaken* attribute of the *Student* class has a set value of 1-20 elements, each of which is 5 byte long. The *courseTaken* attribute of the *Graduate* class has a set of 201-250 elements, each of which is 5 byte long again. In both cases, the values are systematically populated in such a way that we are able to select any portion of the *Student* class and the *Graduate* class by posing appropriate values for the set attribute at query conditions.

In our schema, there are six spatial attributes for GIS applications: the *trail* attribute and *area* attribute of the *College* class, the *c_room* attribute of the *Course* class, the s_*room* attribute of the *Section* class, and the *home* attribute and *office* attribute of the *Professor* class. All values of the spatial attributes are located in a 100 x 100 area. As for the *area* attribute of the polygon data type, we create instances of house shape and of star shape. There are 10 houses and 10 stars for the polygon data type in the 100 x 100 area. The values of the *area* attribute are mapped onto one of the twenty polygons in a circular fashion. The values of the *c_room* and *s_room* attributes are mapped onto one of 2 x 1 and 2 x 2 shape boxes, respectively,

which overlap with each other. The values of the *home* and *office* attributes of the *professor* class are populated such that professors who have an office and a home within 1.1 distance unit constitute 1% of all professors.

The BORD benchmark provides scaleability by use of a scale factor. The scale factor allows the user to generate the test database to reflect a potential application environment appropriately in terms of the database volume. A nominal database size, which is shown in Table 1, is given to the user, and the user is supposed to set the scale factor appropriately. The scale factor determines the number of instances of each class, which is be populated accordingly by a database generator module, a part of the BORD benchmark program.

Table 1. Nominal size and scale factor

Class	Number of instances	Class	Number of instances
Student	100,000 * Scale	*Professor*	30,000 * Scale
Graduate	60,000 * Scale	*External*	40,000 * Scale
Department	500 * Scale	*Fulltime*	20,000 * Scale
College	100 * Scale	*Parttime*	10,000 * Scale
Section	60,000 * Scale	*TA*	50,000 * Scale
Course	30,000 * Scale	*RA*	50,000 * Scale

4 Test Queries

BORD provides a number of test queries that are grouped into categories, based on features of ORDBs. Presently, there are fifteen categories under which thirty-six test queries are defined. For each query, verbal description and an equivalent (pseudo) SQL statement are given.

(1) Exact matches

Retrieval of an instance with an exact match condition over one class and over a class hierarchy is not only a rudimentary function of an ORDB, but also serves as a good baseline of benchmark. Note that the depth of the class hierarchy in the BORD benchmark is seven.

Q1-1 Exact match over a single class: Retrieve the name, state, city, zip, age, and gender of the graduate (the *Graduate* class) with SSN 120000050.

```
Select name, state, city, zip, age, gender
from only(Graduate) where SSN = 120000050;
```

Q1-2 Exact match over a class hierarchy: Retrieve the name, state, city, zip, age, and gender of the person (the class hierarchy rooted at the *Person* class) with SSN 120000050.

```
Select name, state, city, zip, age, gender
from Person where SSN = 120000050;
```

The values of the *SSN* attribute are unique in the test database, hence only one object is satisfied by the search condition. The search conditions of Q1-1 and Q1-2 are intentionally designed to be the same. Comparison of Q1-1 and Q1-2 also brings out explicitly performance difference between when an object is searched in a single class and when the same object is searched over eight different classes forming a class hierarchy of depth seven.

(2) Object identifiers (OIDs)

Support of OIDs is mandatory in ORDBs. An ORBD should make notion of an OID visible to users, so that users can store OIDs at application programs and formulate a query with use of OIDs. We also evaluate a case in which a set of OIDs is returned to the users.

Q2-1 Single OID fetch: Retrieve the object identifier of the chairperson who is in charge of the department with an identifier 10.

```
Select chairperson from Department where departmentId = 10;
```

Q2-2 Fetch of a set of OIDs: Retrieve the object identifiers of the courses that are offered by the department with an identifier 10.

```
Select offers from Department where departmentId = 10;
```

(3) Joins

Join operations remain to be useful in ORDBs, and they affect overall system performance significantly. Here, we are primarily focused on join operations that take place along with a class hierarchy.

Q3-1 Class hierarchy join with no condition: Retrieve the SSN of the student (the class hierarchy rooted at *Student*), and the department name in which he/she majors.

```
Select SSN, D.name from Department D, Student
where departmentId = major;
```

Q3-2 Class hierarchy join with conditions (1% selectivity): Retrieve the SSN of the student (the class hierarchy rooted at *Student*) who lives in a city, ‘city25’, and the department name in which he/she major.

```
Select SSN, D.name from Department D, Student
where departmentId = major and city = ‘city25’;
```

Q3-3 Class hierarchy join with conditions (10% selectivity): Retrieve the SSN of the student (the class hierarchy rooted at *Student*) who lives in a state, ‘S5’, and the department name in which he/she major.

```
Select SSN, D.name from Department D, Student
where departmentId = major and state = ‘S5’;
```

Since the *city* and *state* attribute have 1% and 10% distinct value distribution respectively, the attached conditions prune off the search space of join operations by the factor of 100 or 10, respectively. We believe that comparison of Q3-1, Q3-2, and Q3-3 reveals effectiveness of indices that are created over a class hierarchy, in terms of join operations.

(4) Class references as a domain of an attribute

ORDBs should allow a domain of an attribute of a class to be reference to other classes. In this case, DBMSs usually store OIDs of objects of referenced class as values of the attribute. This feature of ORDBs allows the users to expedite a navigational access in a nested object. Four queries have been defined, depending on selectivity and the number of references.

Q4-1 One-hop reference (1% selectivity): Retrieve the SSN of the TA who lives in a city 'city50', and the department name for which he/she works.

```
Select SSN, worksIn->name from TA where city = 'city50';
```

Q4-2 One-hop reference (10% selectivity): Retrieve the SSN of the TA who lives in a state 'S5', and the department name for which he/she works.

```
Select SSN, worksIn->name from TA where state = 'S5';
```

Q4-3 Two-hop reference (1% selectivity): Retrieve the SSN of the TA who lives in a city 'city50', and the name of the chairperson who manages the department for which he/she works.

```
Select SSN, worksIn->chairperson->name from TA
where city = 'city50';
```

Q4-4 Two-hop reference (10% selectivity): Retrieve the SSN of the TA who lives in a state 'S5', and the name of the chairperson who manages the department for which he/she works.

```
Select SSN, worksIn->chairperson->name from TA
where state = 'S5';
```

(5) Set-valued attributes

There are three different data types of set-valued attributes, namely a (regular) set, a multiset, and a list. Here we evaluate a set type only, since a set is the most representative one among them and basic manipulation mechanisms for the three types are similar.

An important technical issue associated with a set type is how a DBMS stores the set internally. Elements of a set can be stored with other values of other attributes in a physically adjacent space (say, a disk page). Alternatively, they can be stored in a separate file and be accessed in an indirect way possibly in aid of foreign key relationship. The first approach is called in-line sets [12]. With a set with small number of elements, the first approach would be faster to access than the second one. We have defined two kinds of sets, i.e. small set and large set, which differ only in terms of the number of elements. The small set contains 1-20 elements, each of which is 5 byte long. The large set has 201-250 elements, each of which is 5 byte long.

We have defined four queries. The Q5-2 and Q5-4 queries, which retrieve 50% of objects of the class mentioned in the "from" clause, will give credit to ORDBs that are able to implement an in-line set when a set is relatively small. The Q5-1 and Q5-3 queries, which return 10% of objects, will give an advantage to ORDBs that support indices to be defined over set attributes.

Q5-1 Small sets (10% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the student who has taken the course 'saaaa'.

```
Select SSN, name, state, city, zip, age, gender
from only(Student) where CourseTaken ⊇ {'saaaa'};
```

Q5-2 Small sets (50% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the student who has taken the following five courses (i.e. 'gaaaa', 'haaaa', 'iaaaa', 'jaaaa', and 'kaaaa').

```
Select SSN, name, state, city, zip, age, gender
from only(Student)
where CourseTaken ⊇ {'gaaaa', 'haaaa', …, 'kaaaa'};
```

Q5-3 Large sets (10% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the graduate who has taken the course 'febaa'.

```
Select SSN, name, state, city, zip, age, gender
from only(Graduate) where CourseTaken ⊇ {'febaa'};
```

Q5-4 Large sets (50% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the graduate who has taken the following ten courses (i.e. 'gbcaa', 'hbcaa', 'ibcaa', 'jbcaa', 'accaa', 'bccaa', 'cccaa', 'dccaa', 'eccaa', and 'fccaa').

```
Select SSN, name, state, city, zip, age, gender
from only(Graduate)
where CourseTaken ⊇ {'gbcaa', 'hbcaa', …, 'fccaa'};
```

(6) User-defined methods

A method is a function that applies to each instance of a class, performing computation based on the values of attributes of classes. A method belongs to a class and is subject to inheritance in a class hierarchy. Because a method may be executed at the client side and/or at the server side in the client/server architecture, a query optimizer should generate an execution plan that minimizes data transfer between the server and the client. Moreover, method definitions are usually stored at external files that should be, on invocation, loaded and be bound into DBMSs. Method definitions may be pre-computed for fast execution. We have taken into account four aspects relevant with execution of the method: location of execution of methods, complexity of methods, a single class versus a class hierarchy, and method inheritance.

The BORD schema has two different methods with the same name, *salary*. The *salary* method in the *Employee* class is intended to be simple and is defined as "*rate* * *C*", where *C* is a predetermined constant. The *salary* method in the *TA* class is designed to be complex and is defined as below:

$$\begin{aligned}\text{salary} = (base * (age / AGELEVEL))^{BASESALARY} + ceil((SALARYCONSTANTTA * hoursPerWeek * rate) \\ - floor(\sqrt{insuranceFee + basesalary * INSURANCERATE})) \quad .\end{aligned} \tag{1}$$

where *basesalary* is assigned, depending on the *age* value, to each object, and identifiers in upper cases denote some constants. This *salary* method is inherited down along the class hierarchy rooted at the *Employee* class, except for the *TA* class that has its own method with the same name.

Q6-1 Methods over a single class: Retrieve the SSN, name, state, city, zip, and age of the male TA whose salary is greater than or equal to 2,500,000.

```
Select SSN, name, state, city, zip, age from only(TA)
where gender = 'M' and salary(TA) ≥ 2,500,000;
```

Q6-2 Methods in a "select" clause over a single class: Retrieve the SSN, name, state, city, zip, age, and salary for the male TA.

```
Select SSN, name, state, city, zip, age, salary(TA)
from only(TA) where gender = 'M';
```

Q6-3 Methods over a class hierarchy: Retrieve the SSN, name, state, city, zip, and age of all male employee whose salary is greater than or equal to 2,500,000.

```
Select SSN, name, state, city, zip, age from Employee
where salary(Employee) ≥ 2,500,000 and gender = 'M';
```

(7) Built-in versus user-defined operators

ORDBs should allow users to construct user-defined operators, and at the same time user-defined operators may be bound with existing operators if the users wish to do. This capability for functions and operators increases the complexity of the query optimizer greatly. ORDB query optimizers does not understand the semantics of user-defined operators normally, so ORDBs should provide some ways by which the users specify properties of user-defined operators (e.g. negation, commutativity, selectivity, access method, and so on). Query optimizers may utilize such informative properties in the phase of query optimization. We have defined three queries that are syntactically different but semantically equivalent. All of the three queries retrieve 1% of all TAs.

Q7-1 to Q7-3 (1% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the TA whose age is greater than or equal to 60.

Q7-1: Built-in operators.

```
Select SSN, name, state, city, zip, age, gender from only(TA)
where age ≥ 60;
```

Q7-2: User-defined operators.

```
Select SSN, name, state, city, zip, age, gender from only(TA)
where age >> 60;
```

Q7-3: Negated user-defined operators.

```
Select SSN, name, state, city, zip, age, gender from only(TA)
where not (age << 60);
```

Here, the symbols, >> and << , are user-defined operators whose semantics are 'greater than or equal (i.e. ≥)' and 'less than (i.e. <) ', respectively. Comparison of Q7-1 and Q7-2 shows query processing effectiveness between a built-in operator and a user-defined operator. Also, comparison of Q7-2 and Q7-3 shows query processing effectiveness of negation and commutativity of a user-defined operator.

(8) Flattening queries over a single class

A method in an ORDB may be specified either in a host programming language or in SQL or as a combination of the two. If a method is written in SQL and a query regarding the method is issued, a query optimizer of DBMS, which generates a query execution plan, should make use of the method definition to achieve global optimization [12]. Global optimization is likely to find a very optimized plan, because it often identifies redundant part of execution from a global viewpoint and takes care of it suitably. The global optimization strategy is simply impossible if methods are specified in programming languages.

Q8-1 Flattening a complex query over one class: Retrieve the name, state, city, zip, age, and gender of the male graduate with SSN 120000050.

```
Select name, state, city, zip, age, gender
from malegraduate() where SSN = 120000050;
```

Here, the method *malegraduate()* returns all male graduates from the *Graduate* class. The method, which must be written in SQL, looks like below:

```
Select * from only(Graduate) where gender = 'M';
```

If a system is smart enough to flatten the query and carry out global query optimization, then the system searches the *Graduate* class with an index on the *SSN* attribute first. This is because the *SSN* is the primary key of the class, and the search with the *SSN* number returns only one instance, which is then examined to see if the returned one is male. If a system starts to execute the method *malegraduate()* in the beginning, the method returns the half of the graduate students, only one of which could become the final answer. Q8-1 may be executed as fast as Q1-1.

(9-11) Generic abstract data types (ADTs)

DBMSs come with a number of built-in data types (for example, integer, char, float, etc). These built-in data types not only state how the data are stored internally, but they also imply available operations on data implicitly. An ADT is a data type that is created by the user. The user may specify available operations on an ADT he/she has defined, if necessary.

In the literature (as well as in the commercial systems), we have identified two kinds of generic ADTs, an atomic ADT and a composite ADT. The atomic ADT, from the point of view of the user, is atomic, although internally it may be represented by distinct components. The internal organization of an atomic ADT is completely encapsulated: the only way to access the atomic ADT can be made with operators that are also defined by the user. DBMSs supporting the atomic ADT should provide mechanisms that allow the user to define possible operators as he/she wishes. On the other hand, the composite ADT consists of a number of components, each of which may be a built-in type, an atomic ADT if available, or another composite ADT. The composite ADT is analogous to the struct in C or the class in C++. Each member of the composite ADT can be accessed individually.

In the BORD schema, the *c_project_t* data type of the *Professor* and *RA* classes is the complex ADT that consists of four built-in types (an integer type, a char(20) type, a date type, and a date type). The *p_project_t* data type of the *RA* class is a collection type (here, a set) of a char(20) type. The *rank_t* and *time_t* data types of the

Professor class are atomic ADTs, and so is the *time_t* data type of the *RA* class. DBMSs are not expected to understand atomic ADTs, so the user should provide necessary operators for them. Here, six basic comparison operators (namely, =, ≠, >, ≥, <, and ≤) and I/O operators should be provided manually in the form of user-defined methods.

Three categories have been designed for the generic ADTs. Category 9 is for accessing generic ADTs with a simple predicate, category 10 measures simple operations on the generic ADTs, and category 11 is devoted to join operations between generic ADTs. Each category has two test queries, one for an atomic ADT and one for a composite ADT, making six queries all together.

Q9-1 Atomic ADT access (1% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the RA who has a meeting ending at 2:30 p.m.

```
Select SSN, name, state, city, zip, age, gender from RA
where meetingEndTime = '14:30';
```

Q9-2 Composite ADT access (1% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of a professor who has a current project with number 10.

```
Select SSN, name, state, city, zip, age, gender
from Professor where currentProject.projectId = 10;
```

At the *where* clause of Q9-2, *projectId* is the first field name of the composite ADT, *c_project_t*. Q10-1 and Q10-2 use user-defined operations that should be declared in advance. The *elepsedTime* operator of Q10-1 takes two arguments of the *time_t* type, and returns difference of the given times. The *projectTerm* operator in Q10-2 returns an integer value.

Q10-1 Operations on atomic ADT (1% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the research assistant who has a meeting that lasts less than or equal to one hour.

```
Select SSN, name, state, city, zip, age, gender from RA
where elapsedTime(meetingStartTime, meetingEndTime) ≤ 1;
```

Q10-2 Operations on composite ADT (1% selectivity): Retrieve the SSN, name, state, city, zip, age, and gender of the professor who has a current project that runs less than or equal to 365 days.

```
Select SSN, name, state, city, zip, age, gender
from Professor where projectTerm(currentProject) ≤ 365;
```

Q11-1 Join on atomic ADTs: Retrieve the professor name and RA name in which the counsel time of the professor coincides with the end time of meeting of the RA.

```
Select P.name, R.name from Professor P, RA R
where P.counselTime = R.meetingEndTime;
```

Q11-2 Join on composite ADTs: Retrieve the names of the professor and the RA in which their current projects are the same.

```
Select P.name, R.name from Professor P, RA R
where P.currentProject.projectId =
R.currentProject.projectId;
```

(12-15) ADTs for GIS

Category 12-15 considers spatial data types and operations for GIS applications. The BORD schema has four spatial data types: point, line, box, and polygon. We have categorized spatial data requirements into 4 groups as follows;

- Category 12: Spatial topological predicates only (Q12-1, Q12-2)
- Category 13: Spatial and non-spatial predicates (Q13-1, Q13-2)
- Category 14: Geometric operations and functions (Q14-1)
- Category 15: Spatial join (Q15-1, Q15-2, Q15-3)

In total, 8 queries are designed. The details of queries are omitted because of space restriction.

5 Implementation and Measurement

The BORD benchmark has been implemented with two commercial ORDBs on the market. The main purpose of this implementation was to show the feasibility of BORD, not to make an attempt to compare the performance of commercial systems. Since we do not want to release which ORDBs were used, we will denote them as system X and system Y.

Test database instances were generated using a bulk loading facility supported by systems X and Y. First, we generated, for each class, one or more ASCII plain files in which all data were sorted by the primary key attributes in ascending order. These files followed the data population restrictions BORD enforces (for example, the *city* attribute has a 10% unique value distribution). Note that each class has a char(20) type *name* attribute. The *name* attribute was generated with the aid of a random number generator; hence it contained arbitrary string values. Next, each of the ASCII files was sorted by the name attribute with the Unix *sort* command, to shuffle the ASCII files completely. The shuffled data was then loaded into the ORDBs.

We created indices for every attribute and its combinations mentioned in the *where* clause in the BORD test queries. Attributes on which indices were created were not only of regular data types (such as integer, string, etc.), but also of non-traditional data types. Indices on set-based attributes, OIDs, spatial attributes, and even user-defined methods were created as long as they were supported by the ORDBs under consideration. Clustered indices were also used to give potential advantages to ranged queries. Index schemes were not confined to B-tree variations. R-tree indices and others were created if supported. The scope of the indices was not necessarily a single class. If possible, indices on a hierarchy of classes were created. Hierarchical indices are useful for queries involving a hierarchical search with some search predicates.

Application programs were written with ESQL/C or with (proprietary) SQL call level interface (CLI). We did not use a low-level interface such as an application-programming interface, even though this could lead to improved performance results. All interfaces we used were essentially SQL level. We stored all databases in the regular Unix file system, not a raw disk. In order to simulate the cold start environment, we read a huge file into the main memory between query executions to

flush the database buffer. In most cases, the flushing functioned properly. When it did not, we restarted the system.

We counted the number of returned instances of the executed queries for verification. For most of the test queries, we compared the number of instances actually returned with the expected number of instances in order to verify the execution of the test queries. Such verification was possible in the BORD benchmark only because we generated the test database in a very controlled manner. The use of synthesized databases in the benchmark pays off in this matter.

The hardware we used was Sun Sparc 20 with 96MB of main memory, running the Sun Solaris 2.5.1 operating system. A 32MB shared memory was used when necessary. The elapse time was measured using the Unix *gettimeofday* command. We measured the performance of system X and Y with a scale factor one. Appendix A shows the queries and results at a glance.

6 Closing Remarks

We have presented a new benchmark, BORD, for ORDBs. The design philosophy, database schema, database instances, and test queries have been described. An implementation effort of BORD with two commercial DBMSs has been made, and the experimental results were shown. Unfortunately, it was not possible to implement some of categories at this point, because the DBMSs we used did not support functions necessary to implement the test queries. This was also partly because we did not add missing functions manually in our application programs. As a matter of fact, there was a commercial system that did support almost all the BORD queries, when we started to design BORD. However, the DBMS just disappeared from the market, and we could not extend our original site license any more. We believe that in the near future we will find on the market a DBMS that supports most of the BORD queries.

The BORD schema contains various kinds of data distributions so that a query with certain selectivity (for example, 1%, 10%, and 50%) is easily formed. All these distributions in the BORD are uniform; all values are evenly distributed even though they are randomly stored internally. Neither skewed distributions nor normal distributions are embodied in the BORD benchmark. A number of various mathematical distributions, which often better reflect the real world than uniform distributions do, should be considered and will be adopted in the next version of the BORD.

Multimedia data such as images, sounds, and video clips are commonly handled by modern DBMSs. Also, the demand of the users for a capability of processing a large amount of unstructured or semi-structured documents in the context of database technology is becoming ever important partly due to the popularity of the Web. Those features are not part of the object-relational data model, yet they should be supported by competitive modern DBMSs. An extension of the BORD benchmark toward multimedia data and full-text retrieval remains as future work.

References

1. T. L. Anderson, A. J. Berre, M. Mallision, H. Porter and B. Schneider: The HyperModel Benchmark. Proc. of the Int. Conference on Extending Database Technology (1990) 317-331
2. M. Asgarian, M. J. Carey, D. J. DeWitt, J. Gehrke, J. F. Naughton, and D. N. Shah: The BUCKY Object-Relational Benchmark. Proc. of the ACM SIGMOD Conference (1997)
3. M. J. Carey, D. J. DeWitt, and J. F. Naughton: The OO7 Benchmark. Proc. of the ACM SIGMOD Conference (1993) 12-21
4. M. J. Carey, D. J. DeWitt, C. Kant, and J. F. Naughton: A Status Report on the OO7 OODBMS Benchmarking Effort. Proc. of the ACM OOPSLA Conference (1994)
5. R. G. G. Cattell and K. Skeen: Object Operations benchmark. ACM Transactions on Database Systems 17(3) (1992) 1-31
6. J. Gray: The Benchmark Handbook, second edition. Morgan Kaufmann (1993)
7. J. M. Hellerstein and M. Stonebraker: Predicate Migration: Optimizing Queries with Expensive Predicates. Proc. of the ACM SIGMOD conference (1993) 267-276
8. W. Kim: Completeness Criteria for Object-Relational Database Systems. UniSQL Inc. (1996)
9. P. O'Neil: Database Performance. In: A. Tucker (eds.): The Computer Science and Engineering Handbook. CRC Press (1997) 1078-1092
10. H. Schreiber: Justitia – A Generic Benchmark for the OODBMS Selection. Proc. of the Fourth International Conference on Data and Knowledge Systems for Manufacturing and Engineering (1994) 324-331
11. M. Stonebraker, J. Frew, K. Gardels, and J. Meredith: The SEQUOIA 2000 Storage Benchmark. Proc. of the ACM SIGMOD Conference (1993) 2-11
12. M. Stonebraker: Object-relational DBMSs. Morgan Kaufmann (1996)

Appendix: Results at a glance

Query No.	System X	System Y	Query No	System X	System Y	Query No	System X	System Y
Q1-1	1.6	2.6	Q4-4	N/S	38.3	Q7-3	N/S	N/S
Q1-2	1.9	45.1	Q5-1	N/S	99.0	Q8-1	N/S	N/S
Q2-1	N/S	2.7	Q5-2	N/S	136.9	Q9-1	10.5	N/S
Q2-2	N/S	2.7	Q5-3	N/S	2598.0	Q9-2	6.64	15.8
Q3-1	110.3	748.0	Q5-4	N/S	2763.7	Q10-1	10.5	N/S
Q3-2	4.6	6.2	Q6-1	57.8	101.8	Q10-2	2.8	38.7
Q3-3	23.7	33.9	Q6-2	56.8	50.9	Q11-1	Over 2 hrs	N/S
Q4-1	N/S	6.7	Q6-3	116.4	M/L			
Q4-2	N/S	31.9	Q7-1	N/S	N/S	Q11-2	Over 2 hrs	N/S
Q4-3	N/S	7.7	Q7-2	N/S	N/S			

- N/S means "not supported".
- M/L means "memory leak".

An Indexing Structure for Aggregation Relationship in OODB[1]

Xiao Renguo[1], Tharam S. Dillon[1], W.Rahayu[2], E.Chang[2], N.Gorla[3]

[1]The Dept. Of Computing, The Hong Kong Polytechnic University
{csrxiao, csdillon}@comp.polyu.edu.hk
[2]The Dept. of Computer Science & Computer Engineering, La Trobe University, Australia
{rahayu, chang}@cs.latrobe.edu.au
[3]The Dept. of Information System, University of Cincinnati
Nara.Gorla@uc.edu

Abstract. In an object-oriented database, classes are organized according to three types of relationship, namely generalization, association and aggregation relationships. These relationships, which impact the query language in different ways, should be considered if we want to design a suitable indexing structure for an object-oriented database. Many researchers focus on generalization and association relationships when trying to find indexing structures for the OODB. For the aggregation hierarchy, there is no deep research in this field. Though we can use the same indexing architectures as have been proposed for association relationships, we must consider the special features of the aggregation relationship. These particular features of aggregation relationships are discussed in this paper in detail considering both the similarities and differences with the association relationship. Next the paper presents index nesting techniques to support aggregation relationships in composite objects. A comparison between this technique and other indexing techniques is also discussed in this paper.

1 Introduction

In an object-oriented data model, relationships express static inter-object connections, between one object and another and hence capture the structural aspects. Classes can be organized according to three types of relationship, namely generalization, association and aggregation relationships [1,2]. A generalization is a relationship between a general class (called the super-class or parent) and a more specific kind of class (called the subclass or child). It is sometimes called an "is-a" relationship which implies that objects belonging to the child class are specializations of the parent class. These "is-a" relationships among classes are often used to show inheritance between classes. An association is a structural relationship that specifies that objects of one class are connected to objects of another. Given an association connecting two classes, we can navigate from an object of one class to an object of the other class, and vice versa. A plain association between two classes represents a structural relationship between peers, meaning that both classes are conceptually at the same

[1] This research work is supported by The Hong Kong Polytechnic University grant G-V680

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 21–30, 2000.

level, no one more important than the other. An aggregation is a composite (part-of) relationship, in which a composite object ("whole") consists of other component objects ("parts"). This kind of relationship represents a "part-of" relationship, meaning that an object of the whole consists of objects of the part. Aggregation is often treated as a kind of association. But classes between the aggregation relationship are not conceptually at the same level. This feature distinguishes the aggregation relationship from the association relationship. Because the aggregation relationship is used commonly in the areas of engineering, manufacturing and graphics design, we distinguish this special kind of relationship from the association relationship in general meaning.

It is well known that an index is a data structure that helps to organize the data by mapping a key value onto one or more records containing the key value, thus providing a mechanism to efficiently identify the storage location of records [3]. Balanced trees such as B+-trees are popular for indexing structures. The advent of object-oriented database systems (OODBS) has introduced new indexing issues due to the semantic richness of the object-oriented model. The three relationships mentioned above impact the query language along different dimensions.

Because of the inheritance relationship, a query issued against a class may include the instances of its subclasses. This relationship requires indexing techniques supporting queries along inheritance hierarchies. Most of those techniques have been defined as extensions of the class-hierarchy index which is based on B+ trees and involves maintaining a single index for all classes in the given inheritance hierarchy [3]. Since this paper focuses on association and aggregation relationships, we will not discuss this kind of indexing technique in detail here.

In association relationships, a query issued against a class may contain predicates against the nested attributes of the class. These predicates are often referred to as nested predicates. Their evaluation requires navigating along object references, thus performing implicit joins among objects. This group of techniques deals with the efficient evaluation of nested predicates. They are based on pre-computing implicit joins along association hierarchies. The main difficulty within these techniques is the efficient execution of updates. The reason for this is that whenever an update is performed some of the pre-computed joins become invalid and must be recomputed. Bertino and Kim first presented three index structures: nested index, path index and multi-index, which have been later extended to handle inheritance of classes appearing in a path expression [4]. The nested index structure facilitates associative search and update by storing together the key values of the tail attribute and the objects of the head class and the intermediate objects of a path expression in primary records. An auxiliary index, which basically keeps direct reference information between objects, together with extra information in the primary records is used to propagate updates. Choenni etc al. proposed an optimal index configuration by splitting a long path expression into shorter ones, and by indexing the shorter paths with a selected indexing structure [5].

The same thing happens in the aggregation hierarchy structure for the "part-of" relationship of composite objects. For the aggregation hierarchy, there is no deep research in this field. Though we can use the same indexing architectures as have

been used by the association hierarchy, we must consider the special features of the aggregation relationship. These particular features of the aggregation relationship are discussed in the following section.

2 Association and Aggregation: Similarity and Difference

From the definition of association and aggregation in section 1, we note that aggregation is also a special kind of relationship. Sometimes, aggregation is a tightly coupled form of association [2]. In the references [10,11,12], the differences and similarities between aggregation and association are discussed in detail, we briefly summarize the differences in this section. In an aggregation relationship, in which one "whole" can have many "parts" associated through a "part-of" relationship, the entire "part-of" relationship is viewed as one composition not several association relationships. For the database retrieval operation, because the relationship between the "whole" and the "part" is very clearly designated in aggregation relationships, we should be able to retrieve all the aggregation "parts" that belong to a "whole" by identifying the "whole" only. For example, when a "PC" object is accessed, the aggregation "parts" such as "Mainboard", " CPU", "Hard Disk", "CD-ROM" and "Monitor" that belong to that "PC" can also be identified. This feature of aggregation is different from that of association. Since composite objects have semantics of sharability and dependency, we can divide the composite objects into four types: sharable dependent composite objects, sharable independent composite objects, non-sharable dependent composite objects and non-sharable independent composite objects [7]. Because of these special features of aggregation, it is different from association in the deletion operation for the database. Based on the characteristics of association, deletion of one object in the association relationship does not cause any deletion of other associated objects. But deletion of the "whole" object may cause deletion of the "part" objects depending on the type of aggregation. In addition, the relationship between the "whole" and the "part" can have other special semantics such as ordered composition and homogenous composition. These diverse kinds of aggregation relationships enable us to model the real world more flexibly and powerfully. Take homogenous composition as an example, when the "whole" consists of some "parts" with the same type, we have homogenous composition. This kind of model is flexible enough for further extension or modification to include components of another type. On the other hand, if a homogeneous composite is modeled as an association model, it does not directly support extension.

From the discussion above, we note the difference between association and aggregation. An aggregation is a relationship between a "whole" and its "parts" which forms the "whole". But an association is simply a link between different objects in an application. The distinction must be made clear in the indexing structure design in the OODB not only in the modeling, but also in the implementation. In this paper, we focus on the issues related to implementation.

3. Index Nesting Technique to Support the Aggregation Relationship

In the development of many complex database applications, particularly in computer aided design, office applications and engineering applications, the notion of composition is very important. Aggregated or composite objects are built from other component objects that are also composite objects. In this way, it could form a multi-level aggregation hierarchy. Thus how to access the "whole" and the "part" of the composite object efficiently is a challenge for database design. Many researchers focus on the representation of the composite objects [7,8], but few address the index structures of composite objects. It is known that Access Support Relation (ASR) can support the reference chains created by the association relationships in an OODB efficiently [6]. Index nesting can support the superclass-subclass hierarchy naturally and efficiently [9]. In this paper, we combine the two ideas from ASR and nesting index together to build a nested index to support the aggregated/composite objects efficiently.

3.1 Brief Summary of Index Nesting and ASR

3.1.1 Index Nesting for Inheritance Hierarchy in OODB

The support of the superclass-subclass concept in the OODB makes an instance of a subclass also an instance of its superclass. As a result, the access scope of a query against a class in general includes the access scope of all its subclasses, unless specified otherwise. To support the superclass-subclass relationship efficiently, the index must achieve two objectives. First, the index must support efficient retrieval of instances from a single class. Second, it must also support efficient retrieval of instances from classes in a hierarchy of classes. In [9], the author presents a hierarchical tree called an H-tree. A H-tree structure is maintained for each class of a class hierarchy and these trees are nested according to their superclass-subclass relationships. Because the H-trees are appropriately linked to capture the superclass-subclass relationships naturally, this technique can allow efficient retrievals of instances from a class hierarchy.

3.1.2 Access Support Relation (ASR)

The data structure called Access Support Relation (ASR) was proposed for object databases by Kemper and Moerkotte [6]. Its basic idea is to extract the most frequently traversed reference chains from the object base and maintain them redundantly in separate data structures that are called access support relations. That is to say, the ASR is based on redundantly maintaining frequently traversed objects paths.

The ASR keeps the identifiers of objects connected by attribute relationships in a path expression and can span over the reference chains of the path expression. Several alternatives which include full, canonical, left and right extensions and decomposition of access support relations for a given expression are discussed in detail in [6].

3.2 ASR Structures for the Composite Object

In this section, we use the ASR technique discussed in the above section to support aggregation relationships in a composite object. Though the method here uses ASR in similar way to that used in the association relationships, it stores the redundant information for the aggregation relationship between the "whole" and the "parts" which considers this particular feature of the aggregation relationship in an OODB. Here we discuss three different storage structures for the aggregation relationship in a composite object.

3.2.1 ASR Structure Along a Path Expression

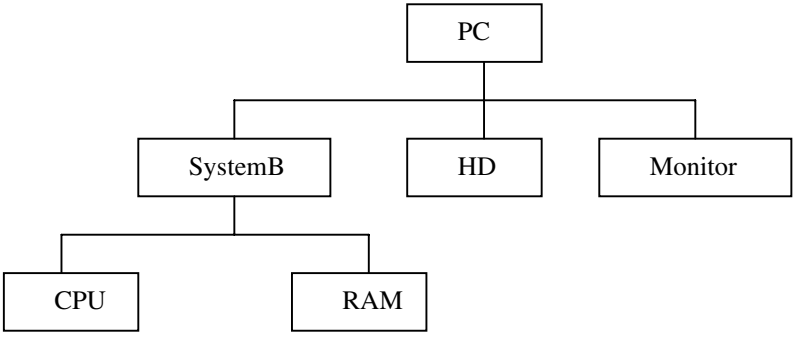


Fig. 1. The Sample Composite Object Schema

OID_{PC}	OID_{SB}	OID_{CPU}
PC1	SB1	CPU1
PC1	SB1	CPU2
PC2	SB2	CPU3
PC2	SB2	CPU4
PC3	SB3	CPU5

Fig. 2. ASR structure along the path PC.SB.CPU

This ASR structure is the same as that used in the association reference discussed in sub-section 3.1. Here we use an example to illustrate how to create the ASR structure along a path expression for the multi-level composite object. This database schema will be used in the following discussion about the other two ASR structures. The object schema is shown in Figure 1. The PC (Personal Computer) consists of a SystemB (System Board), HD (Hard Disk) and a Monitor. The SystemB is also a composite object. It consists of a CPU and a RAM. We use the aggregation model to present the relationships between all these objects. This model has a tree structure for a composite object. Consider a sample composite object PC1 for example, PC1 is composed of SB1, HD1, HD2 and Monitor1. SB1 is composed of two CPUs: CPU1

and CPU2, RAM1 and RAM2. For a composite object, there exists a path expression along the aggregation relationship. In a similar fashion to the path expression in association references, along the relationship between "whole" and "part", we can get a path such as PC.SB.CPU or PC.SB.RAM. Using the ASR technique used in the association relationships in an OODB, we get the ASR structure along the path PC.SB.CPU shown in Figure 2.

3.2.2 ASR Structure for Each Aggregation Relationship

This method considers every aggregation relationship in the composite object separately and creates an ASR structure for each aggregation relationship. Take the database schema in Figure 1 as an example. The relationship between the root class and its direct part classes such as SystemB and Monitor is known as level 1 of path 1. Further down, the relationship between SystemB and CPU+RAM is level 2 of path 1. If Monitor is also a composite object, the relationship rooted by Monitor is level 2 of path 2, etc. For simplicity, we call level x of path y to be AGGREGATIONxy.

Figure 3 shows the represention of aggregation relationships in the database schema. Using the running example shown in Figure 3, the first aggregation structure maintains the relationship between the root class (i.e. PC) and its direct parts (i.e. SystemB, HD and Monitor), whereas the second aggregation structure maintains the Relationship between SystemB and CPU+RAM. Should there be any classes direcly below the class Monitor as shown in Figure 3 as the dashed lines, we will need an aggregation structure to maintain these relationships.

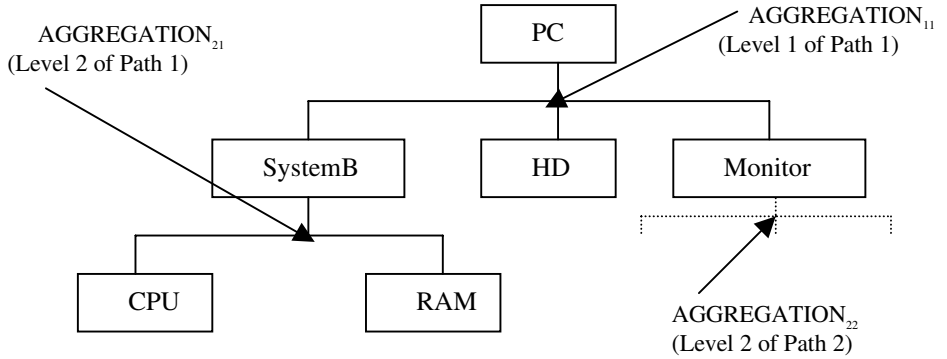


Fig. 3. The represents of aggregation relationships

Using the example above, the type of composition relationship between PC and its parts such as SysytemB, HD and Monitor is a one-one existence independent one. The deletion of particular PC may not necessarily cause a deletion of its components as another PC can reuse them. $AGGREGATION_{11}$ and $AGGREGATION_{21}$ shown in Figure 4 are created to maintain this relationship.

After we get the composite aggregation index structure shown in Figure 4, like the storage model of ASR, we can use two B-trees (both on the left and right) to facilitate the efficient evaluation of forward and backward queries over the composite object.

$Aggregation_{11}$

OID_{PC}	OID_{parts}
PC1	SB1
PC1	M1
PC1	HD1
PC1	HD2
PC2	SB2
PC2	M2
PC2	HD3
PC3	SB3
PC3	M3
PC3	HD4

$AGGREGATION_{21}$

OID_{SB}	OID_{parts}
SB1	RAM1
SB1	RAM2
SB1	CPU1
SB2	RAM3
SB2	RAM4
SB2	CPU2
SB2	CPU3
SB3	RAM5
SB3	CPU4

Fig. 4. ASR structure for each aggregation relationship

3.2.3 Nested ASR Structure

From the ASR structures in the above sub-section, we can create a composite index for the whole composite object. We perform a left join operation to all the aggregate indexes created in the previous sub-section. Using the previous $AGGREGATE_{11}$ and $AGGREGATE_{21}$ index, we get the following composite index using the left outer join operation. The result is shown in Figure 5.

From this composite index, we can efficiently get all the "parts" of a composite object such as the PC composite object as illustrated. But if we only want to know the "parts" of "part" of the composite object such as "SB" in this example, we also have to get all the information of the whole composite object. If the composite object has a lot of "parts" and the aggregation relationships has more than two levels, this query operation will not be efficient. In this sub-section, we present a nested index to efficiently support these multi-level composite objects.

In this kind of nested index, we build a separate ASR structure as in sub-section 3.2.2, then we use the reference to maintain the multi-level composite relationship. In this way, we can form a natural tree structure of the ASR structures as illustrated in Figure 6.

PC_OID	Part1	Part_Type 1	Part2	Part_Type2
PC1	SB1	SystemB	RAM1	RAM
PC1	SB1	SystemB	RAM2	RAM
PC1	SB1	SystemB	CPU1	CPU
PC1	M1	Monitor	NULL	NULL
PC1	HD1	HD	NULL	NULL
PC1	HD2	HD	NULL	NULL
PC2	SB2	SystemB	RAM3	RAM
PC2	SB2	SystemB	RAM4	RAM
PC2	SB2	SystemB	CPU2	CPU
PC2	SB2	SystemB	CPU3	CPU
PC2	M2	Monitor	NULL	NULL
PC2	HD3	HD	NULL	NULL
PC3	SB3	SystemB	RAM5	RAM
PC3	SB3	SystemB	CPU4	CPU
PC3	M3	Monitor	NULL	NULL
PC3	HD4	HD	NULL	NULL

Fig. 5. Composite index for the PC composite object

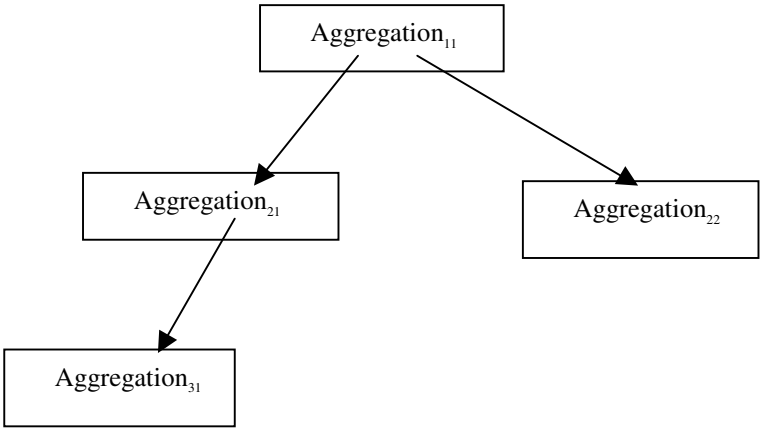


Fig. 6. Illustration of the nested aggregation index

In Figure 6, every Aggregation here is an ASR structure as shown in sub-section 3.2.2. In this structure, we can divide the composite object into different levels of aggregation relationships, hence we can get the exact information when we only want to get the aggregation information of a composite object. In the example of the composite object PC, the Aggregation11 details the information of the relationships between PC and its direct "parts". But for further information of the "parts" of SB, we have to go along the link between $Aggregation_{11}$ to $Aggregation_{21}$ to get the information of the relationship between SB and its "parts". So we have to modify the structure of the ASR as shown in Figure 7. In this figure, when the object in the column of OID_{parts} is also a composite object, we keep a reference to the low-level

aggregation index. For example, Ref_{21} is the reference to the low-level aggregation $Aggregation_{21}$. In this way, the two aggregation indexes form a two-level nested index.

$Aggregation_{11}$

OID_{PC}	OID_{parts}, Ref
PC1	SB1,Ref_{21}
PC1	M1
PC1	HD1
PC1	HD2
PC2	SB2, Ref_{21}
PC2	M2
PC2	HD3
PC3	SB3, Ref_{21}
PC3	M3
PC3	HD4

Fig. 7. Illustration of the nested indexing structure for $Aggregation_{11}$

3.2.4 Comparison of the Above Three ASR Structures

From the above description, we know that all these three ASR structures are based on the ASR technique, but they consider the composite object along different directions. The first ASR structure takes the composite object tree created from the aggregation relationships in the vertical direction and then forms a path expression for this vertical direction from the composite object schema. This structure is suitable for query along the path expression where there is an ASR structure. But because there are many paths along an aggregation relationship, it requires much redundant information for the ASR structures along the paths that exist in a composite object. The second ASR structure considers the composite object in a horizontal direction. It redundantly stores the information between the "whole" and all its direct "parts". In this way, it is suitable for the query along the "whole" and its "parts". But it's not efficient for the queries where there exists a path along a multi-level aggregation relationship and when one wants to know all the "parts" of a composite object, this structure requires a "join" operation among all the separate aggregation indexes. The third ASR structure considers the composite object as a whole and stores all the redundant information of all the relationships between the "whole" and its direct "parts" in a hierarchical structure. This structure requires the least storage, and if we want to know only all the direct "parts" of a "whole", we need not search all the space of the "parts" of the composite object which will affect the query performance.

4. Conclusion

In this paper, we focus on implementation issues related to aggregation relationships in Object-Oriented Databases. Though aggregation is a kind of association, it has its own particular features. Previous research on indexing techniques ignores important differences in the semantics between aggregation and association. This paper discusses all three relationships namely inheritance, association and aggregation in the object-oriented database and how these relationships influence the indexing techniques according to different dimensions. Based on the particular features of the aggregation relationship, three ASR structures are discussed to represent the relationships between the "whole" class and the "part" classes. All these three structures are similar to the Access Support Relation (ASR) structure used to speed up the queries among the reference chain created by association relationships among classes of the OODB. Our future work will focus on the following two aspects: One is to further consider the update and maintenance costs of the three indexing structures for aggregation relationships, hence to find an optimal balance between the cost of queries and updates operations for a given composite object. The other is to take account of different kinds of aggregation relationships such as existence independent and existence dependent.

References

[1] Rumbaugh, J., et. al., Object-Oriented Modeling and Design, Prentice-Hall International, New Jersey, 1991.
[2] Michael Blaha and William Premerlani, Object-Oriented Modeling and Design for Database Applications, Prentice-Hall International, New Jersey, 1998.
[3] Elisa Bertino and Beng Chin Ooi, The indispensability of Dispensable Indexes, IEEE Trans.Knowledge and Data Engineering, vol 11, no. 1, 1999.
[4] E.Bertino and W.Kim, Indexing Techniques for Queries on Nested Objects, IEEE Trans. Knowledge and Data Eng., vol 1, no. 2, pp. 196-214, 1989.
[5] S.Choenni, E. Bertino, H.M. Blanken and T.Chang, On the Selection of Optimal Index Configuration in OO Databases, Proc. Int'l Conf. Data Eng., pp. 526-537, Phoenix, Ariz., 1994
[6] A. Kemper and G. Moerkotte, Access Support in Object Bases, Proc. ACM-SIGMOD Conf. Management Data, pp. 364-374, Atlantic City, N.J., May 1990.
[7] Dillon, T.S and P. L. Tan, Object-Oriented Conceptual Model, Prentice Hall, 1993
[8] Rahayu, W., E. Chang and T.S. Dillon, Implementation of Object-Oriented Association Relationships in Relational Database, Proceedings of International Database Engineering and Applications Symposium, IEEE Press, UK, 1998
[9] Beng Chin Ooi, Jiawei Han, Hongjun Lu, Kian Lee Tan, Index nesting – an efficient approach to indexing in object-oriented databases, The VLDB Journal (1996) 5: 215-228, Springer-Verlag, 1996
[10] Rahayu Wenny, E. Chang and T.S. Dillon, " Composite Indices as a Mechanism for Transforming Multi-Level Composite Objects into Relational Databases", The Object Journal (Special Issue – Best of OOIS'98), Volume 5, No. 1, Hermes Science Publications, 1999.
[11] Rahayu W., Chang E. and Dillon T.S, " Transformation of Object-Oriented Collection Types and their Generic Methods for Implementation in Relational Database Systems", submitted to IEEE Transaction on Knowledge and Data Engineering.
[12] Rahayu W., "Objects Relational Transfer Methodology" Ph.D Thesis, La Trobe University, Australia, 1999

Deciding on the Equivalence of a Relational Schema and an Object-Oriented Schema

Reda Alhajj

Department of Math & Computer Science, American University of Sharjah,
P.O. Box 26666, Sharjah, U.A.E., ralhajj@aus.ac.ae

Abstract. This paper presents a novel approach to decide on the equivalence of a conventional relational schema and a corresponding object-oriented schema. We assume that the two schemas are consistent and implemented a prototype to handle the process. First, characteristics of the relational schema are derived and lead to a graph consistent with the entity-relationship diagram. Second, a graph that include all nesting and inheritance links is derived based on characteristics of the object-oriented schema. Third, we investigate the equivalence of the two graphs in order to decide on relational attributes equivalent to object-oriented attributes with primitive domains, and on the equivalence between relational foreign keys and object-oriented attributes with non-primitive domains.

1 Introduction

The relational schema and the object-oriented schema are the two most popular schemas in recent database applications. It is very common to have the same application served by two information systems, one based on a relational schema and the other based on an object-oriented schema. So, it is necessary to allow accessing the contents of one database from within the realm of another.

Some approaches described in the literature propose an object wrapper that allows relational database reusing, e.g., DRIVER [11], where the user is expected to provide the mapping between the two schemas. Other approaches provide an application development tool that uses an automatic code generator to merge object-oriented programming language (like C++) applications with relational databases, e.g., Persistence [1, 10]. Another related area is semantic enrichment by investigating dependencies [12]. However, most of the work described in the literature concentrates on the re-engineering of the relational schema [6, 9, 13–19]. The latter approaches deal only with one-way mapping.

This paper presents a two-way mapping. It is a novel approach that decides on the equivalence of an existing conventional relational schema and a corresponding existing object-oriented schema. This study is based on our previous research related to object-oriented databases as illustrated in [2, 3]; we also benefited from our on-going research on database re-engineering [4, 5]. Although classes and entities cannot be seen as equivalent, both have common characteristics, including generalization and aggregation. These are the characteristics of

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 31–40, 2000.

major concern to the scope of this paper. Further, it worth mentioning that a major problem in industrial legacy databases is the possibility of a flawed set of tables and relationships. However, the approach described in this paper is not designed to handle this situation; it only focuses on a clean set of tables and relationships. We assume that all relations are in the third normal form and the two schemas are consistent. The consistency constraint necessitates that attributes with primitive domains in a certain class have their equivalent attributes in the corresponding relation, and the values of non-primitive attributes in a class are determined based on the values of the corresponding foreign keys in the corresponding relation, if any.

The importance of this study lies on the fact that it is necessary to get unified global reports that involve information from the two underlying databases. We developed a system and implemented a first prototype that takes the characteristics of the two existing schemas as input, and decides on the targeted equivalence. We examine only the equivalence of the part common to both schemas. In general, it is not necessary to have every relational attribute represented in the object-oriented schema and vice versa. In other words, any of the two schemas might partially or totally cover the other. The best case is to have the two schemas totally overlap, and the worst case is to have the two schemas totally disjoint. Any situation in between is also possible. It is possible to have in the object-oriented schema additional classes/attributes with no corresponding relations/attributes in the relational schema, and vice versa.

The rest of the paper is organized as follows. The basic characteristics of the relational schema are described in Section 2. The basic characteristics of the object-oriented schema are discussed in Section 3. In Section 4, we investigate and decide on the equivalence of the attributes in the two schemas. Section 5 includes the conclusions and future research directions.

2 B s C s s

Consider the relational schema in Example 1, which will be referenced in the rest of the paper where illustrating examples are necessary. As the relational schema is concerned, it is necessary to identify primary and foreign keys of the given relations. Keys related information leads to all unary and binary relationships that exist between the given relations. Each such relationship may correspond to an inheritance or a nesting relationship in the corresponding object-oriented schema.

Example 1 (Relational Schema)

$Person(SSN, name, city, age, sex, SpouseSSN, CountryName)$
$Staff(StaffID, salary, PSSN, DeptName)$
$Prerequisite(CourseCode, PreqCode)$ $Course(Code, title, credits)$
$Student(StudentID, gpa, PSSN, DName)$ $Takes(Code, StudentID, grade)$
$ResearchAssistant(StudentID, StaffID)$ $Country(Name, area, population)$
$Secretary(SSN, words/mimute, DName)$ $Department(Name, HeadID)$ □

The information required to feed the system with the necessary understanding of a given relational schema is summarized by two tables; we named them *ForeignKeys* and *RelationalAttributes*, as illustrated by Table 1*a* and Table 1*b*, which are, respectively, the *ForeignKeys* table and the *RelationalAttributes* table related to the relational schema in Example 1.

Primary Key attributes		Foreign Key attributes		Link #
Relation Name	Attribute Name	Relation Name	Attribute Name	
Person	SSN	Student	PSSN	1
Person	SSN	Staff	PSSN	1
Person	SSN	Secretary	PSSN	1
Person	SSN	Person	SpouseSSN	1
Country	Name	Person	CountryName	2
Student	StudentID	ResearchAssistant	StudentID	1
Student	StudentID	Takes	StudentID	1
Staff	StaffID	ResearchAssistant	StaffID	2
Staff	StaffID	Department	HeadID	1
Course	Code	Prerequisite	CourseCode	1
Course	Code	Prerequisite	PreqCode	2
Course	Code	Takes	Code	2
Department	Name	Student	Dname	2
Department	Name	Staff	DeptName	2
Department	Name	Secretary	DName	2

(a)

Relation Name	Attribute Name	Domain
Person	SSN	integer
Person	name	string
Person	age	integer
Person	sex	character
Country	Name	string
Country	area	integer
Country	population	integer
Student	StudentID	integer
Student	gpa	real
Staff	StaffID	integer
Staff	salary	integer
Course	Code	integer
Course	title	string
Course	credits	integer
Takes	grade	string
Department	Name	string
Secretary	words/minute	integer

(b)

Table 1. a) *ForeignKeys*: Attributes in primary keys and corresponding foreign keys
b) *RelationalAttributes*: Non-foreign key attributes and their respective domains

The *ForeignKeys* table includes all pairs of attributes such that the first attribute is part of the primary key in a certain relation and the second attribute is part of a corresponding foreign key, a representative of the first attribute within any of the relations (including the relation of the primary key itself). *Link#* is there to classify and differentiate attributes that constitute each foreign key. Foreign keys are numbered within each relation such that all attributes in the same foreign key are assigned the same link number. The *RelationalAttributes* table includes information about all attributes that are not part of any foreign key in the relational schema. In the current implementation, the two tables are derived by analyzing the contents of the given relational database. Finally, the information included in *ForeignKeys* is sufficient to decide on all possible links between relations in the given relational schema.

2.1 The Relational Graph

In this section, we use the information present in the *ForeignKeys* table (see Table 1*a*) to draw what we call the *Relational Graph* (*RG*), which includes all possible relationships between the relations present in the given relational schema. Nodes in *RG* are relations and two nodes are connected by a link to show that a foreign key in the relation that correspond to the first node represents the primary key of the relation that correspond to the second node. Nodes and

links are represented by small rectangles and directed arrows, respectively. More formal details related to RG are included in Definition 1, given next.

Definition 1 (Relational Graph).
Every relational schema has a corresponding RG graph (V, E), where

1. V includes every relation R that appears in *ForeignKeys*.
2. Let R_1 and R_2 be two nodes in V, and let l be a score in the range $[0, 2]$. An edge (R_2, R_1, l) is added to E if and only if there exists a tuple (R_1, x, R_2, y, i) in *ForeignKeys*, such that x is an attribute in R_1, y is an attribute in R_2, and i is Link#. The edge (R_2, R_1, l) is directed from R_2 to R_1 and only one edge (R_2, R_1, l) is added to E for every primary key and foreign key pair. □

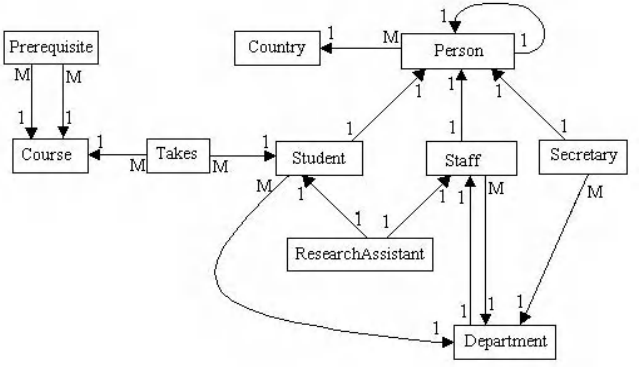


Fig. 1. The Relational Graph of the relational schema in Example 1

Shown in Figure 1 is the RG graph derived from the information present in Table 1*a*. Notice that, two links are directed from *Prerequisite* to *Course* because the primary key of the *Course* table has two corresponding foreign keys in the *Prerequisite* table. Links in RG are classified according to their cardinalities into two groups, 1:1 and M:1. This classification is based on the definition of RG where a link is directed from R_2 to R_1 to reflect the presence of the primary key of R_1 as a foreign key in R_2. The cardinality of a link or relationship is 1:1 if and only if at most one tuple from R_2 holds the value of the primary key of a tuple from R_1. Otherwise, the cardinality is classified as M:1. Some 1:1 links are actually subtyping links.

Sub-Relation	Super-Relation
Student	Person
Staff	Person
Secretary	Person
ResearchAssistant	Student
ResearchAssistant	Staff

Table 2. *Generalization*: A list of all possible sub-relations and super-relations

To identify 1:1 links that represent subtyping, what we call the *Generalization* table is constructed to include pairs of relation names such that the first relation is a sub-relation of the second. Tuples are included in *Generalization* by

classifying links based on the contents of *ForeignKeys*. A link is considered as a candidate to be a subtyping link if and only if the foreign key in the link is also a candidate key of its relation. According to this classification, Table 2 is the *Generalization* table related to the relational schema in Example 1. In the current implementation, the system displays all candidate subtyping links and relies on an expert to suggest tuples to be included in *Generalization*.

Finally, links are assigned scores such that only 1:1 links that represent subtyping are given the score 0, all other links get the score 1. Later in section 4, we will show how some links change score from 1 to 2, by considering the information present in *ForeignKeys*.

3 C s s

In this section, we investigate characteristics of the given object-oriented database and as a result derive the object graph. We start by presenting the basic terminology and definitions required for the scope of this paper. We are mainly interested in the following class characteristics as illustrated next in Example 2. Given any class c,

1. $C_p(c)$: is a list of the direct superclasses of class c.
2. $L_{attributes}(c)$: is the set of additional attributes (not inherited) locally defined in class c. Every attribute in a class has a domain, which is formalized in the following definition.

Definition 2 (Domain).
Let c_1, c_2, ..., and c_n be primitive and user defined classes, where primitive classes include reals, integers, strings, etc. The following are possible domains,

1. $(a_1{:}c_1, a_2{:}c_2, ..., a_n{:}c_n)$ is a tuple domain; a possible value is a tuple constructed using object identifiers selected from c_1, c_2, ..., and c_n, respectively.
2. c_i, $1 \leq i \leq n$, is a domain; a possible value is an object identifier from class c_i.
3. $\{d\}$ is a domain, where d may be any of the two domains defined in 1 and 2; a possible value is a set of values from domain d.
4. $[d]$ is a domain, where d may be any of the two domains defined in 1 and 2; a possible value is a list of values from domain d. □

Example 2 (Classes)
Next are characteristics of the classes in the object-oriented schema that correspond to the relational schema in Example 1:

Person: $C_p(Person) = []$
$L_{attributes}(Person) = \{SSN{:}integer, name{:}string, age{:}integer, sex{:}character, spouse{:}Person, nation{:}Country\}$

Country: $C_p(Country) = []$
$L_{attributes}(Country) = \{Name{:}string, area{:}integer, population{:}integer\}$

Student: $C_p(Student) = [Person]$
$L_{attributes}(Student) = \{StudentID{:}integer, gpa{:}real, student\_in{:}Department, Takes{:}\{(course{:}Course,\ grade{:}string)\}\}$

Staff: $C_p(Staff)$=[*Person*]
$L_{attributes}(Staff) = \{StaffID{:}integer, salary{:}integer, works\_in{:}Department\}$
ResearchAssistant: $C_p(ResearchAssistant)$=[*Student*, *Staff*]
$L_{attributes}(ResearchAssistant) = \{\}$
Course: $C_p(Course)$=[]
$L_{attributes}(Course) = \{Code{:}integer, title{:}string, credits{:}integer, Prerequisite{:}\{Course\}\}$
Department: $C_p(Department)$=[]
$L_{attributes}(Department) = \{Name{:}string, head{:}Staff\}$
Secretary: $C_p(Secretary) = [Person]$
$L_{attributes}(Secretary) = \{words/mimute{:}integer, works\_in{:}Department\}$ □

Class Name	Attribute Name	Domain
Person	ssn	integer
Person	name	string
Person	age	integer
Person	sex	character
Country	name	string
Country	area	integer
Country	population	integer
Student	studentID	integer
Student	gpa	real
Staff	staffID	integer
Staff	salary	integer
Course	code	integer
Course	title	String
Course	credits	integer
Department	name	string
Secretary	words/minute	integer
T_1	grade	string

(a)

Class Name	Attribute Name	Domain
Person	spouse	Person
Person	nation	Country
Student	student_in	Department
Student	Takes	T_1
Staff	works_in	Department
Course	prerequisite	Course
Department	head	Staff
Secretary	works_in	Department
T_1	course	Course

(b)

Table 3. *ObjectAttributes*: (a) a list of all attributes with primitive domains (b) a list of all attributes with non-primitive domains

The necessary information related to a given object-oriented schema is summarized in the table: *ObjectAttributes*(*class name*, *attribute name*, *domain*), where for each attribute, it is required to know its name, class and domain. Attributes with primitive domains and attributes with non-primitive domains are placed in separate occurrences of the *ObjectAttributes* table, namely *ObjectAttributes*(a) and *ObjectAttributes*(b), respectively. Table 3 includes the *ObjectAttributes* information related to the object-oriented schema introduced in Example 2. Each domain of the tuple type is assigned a short name that consists of the letter 'T' suffixed with a consecutive non-decreasing number, starting with 1. For instance, as shown in the fourth row in Table 3*b*, the short name T_1 has been assigned to the domain of *Takes* from $L_{attributes}(Student)$. This way, it becomes trivial to identify attributes that appear within a tuple domain as illustrated in the last row in each of *ObjectAttributes*(a) and *ObjectAttributes*(b) in Table 3.

3.1 The Object Graph

In this section, we use the information present in *ObjectAttributes*(b) and the inheritance information to draw what we call the *Object Graph* (*OG*), which includes all possible relationships between the classes present in the given object-oriented schema. Nodes in *OG* are classes and representatives of tuple type domains. Two nodes are connected by a link to show the inheritance or a nesting relationship between them. Nodes and links are represented by small rectangles and directed arrows, respectively. Inheritance and nesting links are assigned the scores 0 and 1, respectively. A link is assigned the score 2 if it is connecting a node that represents a tuple domain and the class in which it is referenced. To illustrate this, refer to the attribute *Takes* in $L_{attributes}(Student)$ in Example 2, and to the corresponding link that connects the two nodes T_1 and *Student* in Figure 2. More formal details related to *OG* are included next in Definition 3.

Definition 3 (Object Graph).
Every object-oriented schema has a corresponding *OG* graph (V, E) such that,

1. For every class c, there is a corresponding node c in V
2. For all classes c_1 and c_2, such that $c_2 \in C_p(c_1)$, an edge $(c_1, c_2, 0)$ is added to E
3. For every class c
   For every attribute $a \in L_{attributes}(c)$, a has a non-primitive domain
   If domain of a involves a class, say c', then
   . An edge $(c, c', 1)$ is added to E
   ElseIf domain of a involves a tuple T_i, $(i \geq 1)$ then
   - A node T_i is added to V and an edge $(T_i, c, 2)$ is added to E
   - For every class c'' that appears as a domain in tuple T_i
   . An edge $(T_i, c'', 1)$ is added to E □

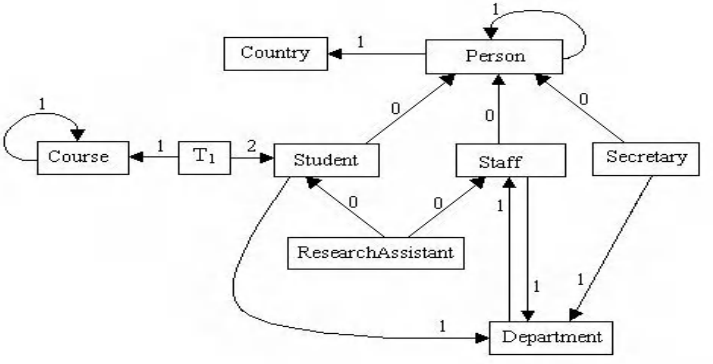


Fig. 2. The Object Graph of the object-oriented schema in Example 2

As Definition 3 is concerned, every node T_i in V, where an edge $(T_i, c'', 2)$ exists in E, corresponds in *RG* to a node that represents a relationship with attributes or a relationship that involves more than two relations. This will be more clear later in Section 4, when the equivalence of the two graphs *RG* and *OG* is investigated. Shown in Figure 2 is the *OG* graph derived from the information present in Table 3*b* and the inheritance information in the C_p lists in Example 2.

4 G p s

In this section, we investigate the equivalence of the two graphs RG and OG. For this purpose, we implemented a first prototype, which first decides on the equivalence between attributes with primitive domains in the object-oriented schema and attributes that are not part of any foreign key in the relational schema. Then, attributes with non-primitive domains in the object-oriented schema are matched with foreign keys in the relational schema. The basic input required for this process consists of the two graphs RG and OG, the three tables *ForeignKeys*, *ObjectAttributes*(a) and *ObjectAttributes*(b), the relational database and the object-oriented database. Contents of the two databases are necessary for cases where multiple choices exist and the user is consulted (by displaying some database contents) to help in the decision making process. The produced output is summarized by the following two tables.

Equivalence(*relation name*, *rel. attr. name*, *class name*, *object attr. name*)

EquivOfForKeys(*class name*, *object attribute name*, *relation name*, *Link#*)

The first table keeps track of primitive attributes in the object-oriented schema and their corresponding attributes in the relational schema. The second table holds a tuple for each non-primitive attribute in the object-oriented schema to show its corresponding foreign key within the relational schema. Since a foreign key may consist of more than one attribute, a foreign key is referenced by its *Link#* from the *ForeignKeys* table.

Relation Name	Attribute Name	Class Name	Attribute Name
Person	SSN	Person	ssn
Person	name	Person	name
Person	age	Person	age
Person	sex	Person	sex
Country	Name	Country	name
Country	area	Country	area
Country	population	Country	population
Student	StudentID	Student	studentID
Student	gpa	Student	gpa
Staff	StaffID	Staff	staffID
Staff	salary	Staff	salary
Course	Code	Course	code
Course	title	Course	title
Course	credits	Course	credits
Department	Name	Department	name
Secretary	words/minute	Secretary	words/minute
Takes	grade	T_1	grade

(a)

Class Name	Attribute Name	Relation Name	Link #
Person	spouse	Person	1
Person	nation	Person	2
Student	student_in	Student	2
Student	takes	Takes	1
Staff	works_in	Staff	2
Course	prerequisite	Prerequisite	1
Department	head	Department	1
Secretary	works_in	Secretary	2
T_1	course	Takes	2

(b)

Table 4. a) *Equivalence*: Object-oriented attributes with primitive domains and their corresponding relational attributes

b) *EquivOfForKeys*: Object-oriented attributes with non-primitive domains and the Link# of their corresponding foreign keys in the relational schema

The process involves a sequence of four major parts. The first part decides on the equivalence of single-valued object-oriented attributes with primitive domains and relational attributes that do not participate in any foreign keys. Nodes in

the two graphs RG and OG are considered starting with those connected to an edge with 0 score. Further, any node v that satisfies this is not considered until for all edges $(v, v', 0)$, the node equivalent to node v' is determined. The second part is dedicated to the equivalence of a tuple domain in the object-oriented schema and relations that represent relationships with attributes and n-ary relationships in the relational schema. Links with score 2 in OG are considered and scores of the corresponding links in RG are changed from 1 to 2. Single valued object-oriented attributes with non-primitive domains are considered in the third part to find their corresponding foreign keys in the relational schema. The latter foreign keys represent 1:1 and 1:M relationships without attributes. Finally, the fourth part checks multi-valued object-oriented attributes and determine their corresponding foreign keys in the relational schema. The first part leads to the *Equivalence* table and the other three parts construct the *EquivOfForKeys* table.

Links in the inheritance hierarchy are given priority over links in the nesting hierarchy. Further, links in the inheritance hierarchy are considered in top-down order. In other words, generalization is given priority over specialization and aggregation. This order is important because a class utilizes the attributes defined for its superclass(es). Therefore, deciding on the relation(s) equivalent to the superclass(es) of any given class simplifies the process of investigating its equivalent relation. Tables 4*a* and 4*b* are the output of the described process with the two graphs shown in Figures 1 and 2 as input. Finally, the change in the scores of some edges in RG from 1 to 2 is necessary to match the scores of the corresponding edges in OG. For instance, the score of the edge connecting the two nodes *Prerequisites* and *Course*, where Link# = 1, is changed from 1 to 2. This change of score leads to consistency between links in RG and their equivalent links in OG, and hence guides the system during the processing of database contents about how to derive the values that constitute objects out of the values that constitute tuples and vice versa.

5 C s s

In this paper, we investigated the equivalence of a relational schema and a corresponding object-oriented schema. As a result of this study, it is possible to utilize values from tuples of the relational database as objects in the object-oriented database and vice versa. The schemas equivalence decision presented in this paper allows accessing the contents of one database within the realm of another database. Only the additional object-oriented functionalities will be hidden when the values in the object-oriented database are mapped to the relational model. On the other hand, the additional features of the object-oriented model can be applied to the contents of the relational database mapped to the object-oriented model.

In the current implementation, the system seeks help from the user in resolving conflicts while deciding on the equivalence between attributes and between links. User help is required to guide the system only in case of multiple choices. We are planning to minimize the dependency on users by providing autonomous

agents that help in conflict resolution. Another point that we are concentrating on, is to investigate how far we can relax the schema consistency constraint imposed in the current system. In other words, we handled in this paper the case of having attributes with primitive domains in a single class and non-foreign key attributes in a single relation equivalent. We will study the effect of having more than one relation holding relational attributes that are equivalent to attributes with primitive domains in a given class. Finally, we are planning to improve the performance by parallelism.

S

1. S. Agarwal, C. Keene and A.M. Keller, "Architecting Object Applications for High Performance with Relational Databases," *Proc. of OOPSLA Workshop on Object Database Behavior, Benchmarks, and Performance,* Austin, TX, Oct. 1995.
2. R. Alhajj and A. Elnagar, "Incremental Materialization of Object-Oriented Views," *DKE,* Vol.29, No.2, pp.121-145, Nov. 1998.
3. R. Alhajj and M.E. Arkun, "A Query Model for Object-Oriented Database Systems," *Proc. of IEEE-ICDE,* pp.163-172, Apr. 1993.
4. R. Alhajj and F. Polat, "Database Reverse Engineering," *Proc. of ISCIS,* Oct. 1999.
5. R. Alhajj, "Documenting Legacy Relational Databases," *Proc. of ER-REIS Workshop,* LNCS, Springer-Verlag, Nov. 1999.
6. M. Andersson, "Extracting an Entity-Relationship Schema from a Relational Database through Reverse Engineering," *Proc. of ER,* pp.403-419, Dec. 1994.
7. T. Barsalou, et. al, "Updating Relational Databases through Object-Based Views," *Proc. of ACM-SIGMOD,* May 1991.
8. M. Blaha, "Dimensions of Relational Database Reverse Engineering," *Proc. of WCRE,* pp.176-183, Oct. 1997.
9. R.H.L. Chiang, T. Barron and V. Storey , "A Framework for the Evaluation of Database Reverse Engineering Methods for Relational Databases", *DKE*, Vol.21, pp.57-77, 1997.
10. A.M. Keller, S. Agarwal and R. Jensen, "Enabling the Integration of Object Applications with Relational Databases," *Proc. of ACM-SIGMOD,* May 1993.
11. F. Lebastard, "Is an object layer on a relational database more attractive than an object database?," *Proc. of the Workshop on Reasoning about Structured Objects: Knowledge Representation Meets Databases,* Bielefeld, Sep. 1995.
12. S. Lopes, J.-M. Petit and L. Lakhal, "Efficient Discovery of Functional Dependencies and Armstrong Relations," *Proc. of EDBT,* pp.350-364, 2000.
13. J. Manfred and U. Johnen, "An Executable Meta Model for Re-Engineering of Database Schemas," *Proc. of ER,* Dec. 1994.
14. W. Meng, C. Yu, and W. Kim, "A Theory of Translating from Relational Queries to Hierarchical Queries, *IEEE TKDE,* Vol.7, No.2, pp.228-245, 1995.
15. S.B. Navathe and M.K. Pillallamarri, "OOER: Toward Making the ER Approach Object-Oriented, *Proc. of ER,* pp.55-76, 1989.
16. J.M. Petit, et al., "Using Queries to Improve Database Reverse Engineering," *Proc. of ER,* pp.369-386, Dec. 1994.
17. W. Premarlani and M. Blaha, "An approach for reverse engineering of relational databases," *CACM,* Vol.37, No.5, pp.42-49, 1994.
18. O. Signore, et al., "Reconstruction of ER Schema from Database Applications: A cognitive Approach," *Proc. of ER,* pp.387-402, Dec. 1994.
19. Z. Tari and J. Stokes, "Designing the Reengineering Service for the DOK Federated Database System," *Proc. of IEEE-ICDE,* pp.465-475, Apr. 1997.

Semantic Reasoning based Video Database Systems

Duc A. Tran, Kien A. Hua, and Khanh Vu

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816, USA.
{dtran, kienhua, khanh}@cs.ucf.edu

Abstract. A constraint of existing content-based video data models is that each modeled semantic description must be associated with time intervals *exactly within* which it happens and semantics not related to any time interval are not considered. Consequently, users are provided with limited query capabilities. This paper is aimed at developing a novel model with two innovations: (1) Semantic contents not having related time information can be modeled as ones that do; (2) Not only the temporal feature of semantic descriptions, but also the temporal relationships among themselves are components of the model. The query system is by means of reasoning on those relationships.
To support users' access, a video algebra and a video calculus as formal query languages, which are based on semantic relationship reasoning, are also presented.

1 Introduction

Increasingly, there are more and more demands for video to be managed by video database systems (VDBMSs), similar to the way alphanumerical data is managed in traditional databases. The most important issue confronting VDBMSs is the description of the structure of video data in a form appropriate for querying, updating, and presentation. To address this, there have been a number of proposals which can be grouped into two categories:

- **Physical feature based modeling** [13, 11, 10, 3, 4, 17]: Much research has been done in the area of video modeling/querying based on audio-visual features, such as audio, color, texture and motion, captured by image processing or computer vision techniques. Those detected features are used as keys for users' retrieval and querying. However, non-expert users might not want to choose color or motion parameters to form a query. Additionally, semantics of a video do not reside only in how the video is built physically, so video data retrieval based on semantic contents seems more natural and preferable to users.
- **Semantic content based modeling**: A crucial property of a video information system is how it handles semantics. Most existing approaches like

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 41–50, 2000.

this, which are very sparse, deal with models of video objects associated with their semantic descriptions. Examples of this group are [7, 9, 6].

The second direction has been dominant in finding semantic foundations for representing and querying video information due to its flexibility and capability of going far beyond the physical characteristics of video data. Semantics based models are in turn classified into two main tracks, *segmentation-based* and *stratification-based* models. The former [3, 5, 8, 15] first segment the video stream into a set of temporally ordered shots and then build a multilevel abstraction upon them. The drawback of this approach, as pointed out in [14], is the lack of flexibility and the incapability of representing semantics residing in overlapped segments. In a contrary direction, stratification models [14, 9, 1, 16, 12] segment contextual information of the video instead of simply partitioning. The video units, each called a stratum, can overlap and encompass each other and are associated with a time interval corresponding to a physical segment in the video stream. Most recently is [7] which extends the conventional stratification concept by allowing an event to associate with multiple time intervals.

In overall, current approaches only focus on capturing a time interval, or a set of time intervals, associated to a given semantic description. Descriptions without any related time tag are not taken into account, hence the scope of questions about the video is very limited. Our work takes into account the importance of *semantics-semantics relationships*. It is also encouraged by the fact that different information extraction techniques bring out various types of knowledge. In the context of video data, such knowledge can be either time information of an event, or temporal relationships among events, or both. It is a good idea to take advantage of all knowledge sources as much as possible in order to best strengthen the VDBMS. We study a video data model, called SemVideo, with the following properties: (1) Some semantic contents not having related time information are modeled as ones that do; (2) Not only the temporal feature of semantic descriptions, but also the temporal relationships among themselves are components of the model.

Based on the model, we derive mechanisms for the query system that provides reasonably powerful capabilities to end-users for their efficiently questioning about the database. In particular, we propose two formal query languages which are a video algebra and a video calculus. They exhibit a comprehensive set of formations working at both video and within-video level.

The rest of this paper is organized as follows. In section 2, we formally present the details of SemVideo. The query languages are described in section 3. Finally, we give some concluding remarks in section 4.

2 Semantic Video Data Model

In this section, we attempt to address the issues from the aforementioned discussion. The new model is called the *Semantic Video Data Model* (SemVideo) because the primary foci are semantic descriptions, which are the meaningful

information about the video, and their mutual relationships. We remove the constraints of previous content-based models and add a new dimension to be managed by the database, which is inter-description relationships.

Our SemVideo model has the following types of information: (1) Videos: The database manages many videos, each being represented by a unique identifier; (2) Video objects: Basically, a video object is nothing but a video sequence. In this paper, it is further extended to be a set of video segments that satisfy some constraint. Video objects are abstract and not really stored in the database; (3) Semantic objects: A semantic object is a description of knowledge about the video. It has a number of attributes, each having a corresponding value. Each semantic object in the video has a unique identifier to differentiate from others; (4) Entities: An entity can be either a video, a video object or a semantic object; (5) Relationships: A relationship is an association between two entities. It can be time related or semantic-related. Note that, in existing models, relationships are limited to only time relationships between video objects, based on which the relationships between semantic objects are established.

Let Ω be the set of all possible (time) **intervals** which can be written as $[t_1, t_2]$ where t_i's are integers and $t_1 \leq t_2$. Here comes the description of the proposed model.

Definition 1. *[Semantic Video Database] A* **semantic video database** *VDB is defined as a 7-tuple* $VDB = (\Sigma, \Delta, \Gamma, f_{DOM}, f_\Gamma, f_\Delta, f_{REL})$
where

- Σ*: Set of videos in the database. Each video is represented by a unique identifier.*
- Γ*: Set of semantic object attributes.* $TIME$ *is an element of this set regarding the time information for a semantic object.* $CONTAIN$ *is an attribute specifying what other semantic objects also happen during the current one happens.* ID *is a mandatory attribute of any semantic object.*
- f_{DOM} *is a function to map an attribute* γ *to its* value domain $f_{DOM}(\gamma)$. *Especially,* $f_{DOM}(TIME) = \Omega$, $f_{DOM}(CONTAIN) = 2^{\aleph}$ *and* $f_{DOM}(ID) = \aleph$. *In the other cases, a value domain can be* $2^{\aleph}$, $2^{\Re}$ *or* $2^{\Im}$ *where* $\aleph$ *is the set of natural numbers,* $\Re$ *the set of real numbers, and* $\Im$ *the set of strings.*
- Δ*: Set of all* possible semantic objects*, each being defined as the following tuple* $\delta = (ID$: soid, γ_1: $value_1$, γ_2: $value_2$, ..., γ_n: $value_n)$ *where soid is the identifier of the semantic object,* $\{\gamma_1, \gamma_2, .., \gamma_n\}$ *is a subset of* Γ *and* $value_i$ *is an element of* $f_{DOM}(\gamma_i)$ *for i from 1 to n.*
- f_Γ*:* $\Sigma \rightarrow 2^\Gamma$ *is a mapping function from* Σ *to the set of all subsets of* Γ. *For each video* σ *in* Σ, $f_\Gamma(\sigma)$ *gives a subset of* Γ, *that is the set of attributes used for the video.*
- f_Δ*:* $\Sigma \rightarrow 2^\Delta$. *For each video* σ *in* Σ, $f_\Delta(\sigma)$ *returns a set of tuples, each being a possible semantic object whose attribute set is a subset of* $f_\Gamma(\sigma)$. *Such tuples are classified as* semantic objects *of video* σ.
- f_{REL} *is a mapping function that, given a video* σ*, returns a time relation of the following form* $f_{REL}(\sigma)$*:* $f_\Delta(\sigma) \times f_\Delta(\sigma) \rightarrow ALLEN \cup \{\text{VOID}\}$ *where*

ALLEN is the set of the thirteen interval relations given in [2]. If the returned value is VOID, the two semantic objects have no time relationship. $f_{REL}(\sigma)$ *is called the* relationship function *for the video* σ.

Note that the SemVideo model encompass the (conventional or extended) stratification model. A distinguishing feature of SemVideo is the allowance of a semantic object not associated with a time interval to be captured and that the relationship among semantic objects (represented by relationship function) is a component of the model. Time information of a semantic object can be computed based on these relationships and already-known temporal information of other objects. Before we go any further, let us give an example of a video database where only one video is examined. The video is part of the movie *Assassins*. To know what this segment of movie is about, consider the following script:

"Tiled roofs, the stark white stucco of a colonial town square. Black iron bars at a bank. A briefcase carried in a man's hand. A sniper's rifle being assembled. Thick blocks of hundred dollar bills. Placed in the briefcase. A man's teeth as he smiles grimly at the sight. The briefcase snaps shut. A vault door slams. Rubber soles walk a tiled floor. Ahead, brilliant, white light suffuses the exit. We're either going outdoors or over to the other side. A long rifle silencer juts from a window. We see the shooter from behind, a view over his shoulder. In the bank, the man crushes out a cigarette. Only the plaza pigeons notice. As they take flight a man lies dead on the cobblestones and as we look up toward the window, there is nothing there. The pigeons wheel above the plaza. We follow, finally losing them to the sky. The sky changes from gray to blue."

Benefiting from knowledge obtained by several extraction techniques, we can build a video database as follows. In our video database, $\Sigma = \{\sigma\}$, $\Gamma = \{$ID, SUBJECT, ACTION, TIME$\}$, $f_\Gamma(\sigma) = \Gamma$, f_{DOM}(SUBJECT) = f_{DOM}(ACTION) = $2^{\Im}$, $f_\Delta(\sigma) = \{\delta_i\}$ for i = 1..15.

δ_1 = (1, SUBJECT: {roof, town square}, TIME: [0, 3])

δ_2 = (2, SUBJECT: {bank, black iron bars})

δ_3 = (3, ACTION: {carry}, SUBJECT: {briefcase, hand}, TIME: [4, ∞])

δ_4 = (4, ACTION: {assemble}, SUBJECT: {rifle, murderer}, TIME: [5, ∞])

δ_5 = (5, ACTION: {place}, SUBJECT: {money, briefcase}, TIME: [0, 7])

δ_6 = (6, ACTION: {smile}, SUBJECT: {teeth, murderer})

δ_7 = (7, ACTION: {shut}, SUBJECT: {briefcase}, TIME: [7, ∞])

δ_8 = (8, ACTION: {slam}, SUBJECT: {vault door})

δ_9 = (9, ACTION: {walk}, SUBJECT: {soles}, TIME: [0, 8])

δ_{10} = (10, SUBJECT: {exit sign})

δ_{11} = (11, ACTION: {jut}, SUBJECT: {murderer, shooter, long rifle, window, shoulder}, TIME: [9, 12], CONTAIN: {12})

δ_{12} = (12, ACTION: {crush}, SUBJECT: {person, cigarette}, TIME: [8, 10] $\sqcup$ [12, 13])

δ_{13} = (13, ACTION: {die}, SUBJECT: {person})

δ_{14} = (14, ACTION: {fly}, SUBJECT: {pigeon}, TIME: [14, 15])

δ_{15} = (15, ACTION: {change color}, SUBJECT: {sky})

$f_{REL}(\sigma)(\delta_1, \delta_2)$ = BEFORE (i.e., δ_1 ends before δ_2 starts)

$f_{REL}(\sigma)(\delta_2, \delta_3)$ = DURING (i.e., they happen during each other)
$f_{REL}(\sigma)(\delta_3, \delta_5)$ = BEFORE, $f_{REL}(\sigma)(\delta_4, \delta_5)$ = DURING
$f_{REL}(\sigma)(\delta_4, \delta_{10})$ = BEFORE, $f_{REL}(\sigma)(\delta_7, \delta_8)$ = BEFORE
$f_{REL}(\sigma)(\delta_8, \delta_9)$ = BEFORE, $f_{REL}(\sigma)(\delta_{11}, \delta_{13})$ = BEFORE
$f_{REL}(\sigma)(\delta_{13}, \delta_{14})$ = IN, $f_{REL}(\sigma)(\delta_{14}, \delta_{15})$ = BEFORE

Note that semantic objects can have TIME information or not and their relations are represented using the relationship function f_{REL} as above.

3 Query Languages

We now come to introducing content-based query mechanisms to support users' access. Given a number of videos, semantics about them are stored in the database using the model. Users can use them to retrieve various information of interest. To set the stage for a detailed look at video queries and query languages, we begin by clarifying what kinds of outputs are allowed in the querying system. We focus on answers of the following types: (1) a Boolean value (2) a video object (3) a semantic object (4) a video.

We introduce two formal query languages associated with SemVideo model, they are *video algebra* and *video calculus*. Prior to that, we need to define the operators to compare among attribute values of semantic objects, and then define what a selection condition is, which is useful for language descriptions later.

Given a semantic object δ, let $\gamma(\delta)$ be the value of attribute γ of δ.

Definition 2. *[Interval Operators]* **I-operators** *(***I** *stands for "interval") are operators applied on intervals. They ($\sqsubseteq$, $\sqsupseteq$, $\nearrow$, $\nwarrow$) are introduced as follows, where p and q are intervals: (1) $p \sqsubseteq q$: any semantic property of p must be true for q (2) $p \sqsupseteq q$: $p \sqsupseteq q$ iff $q \sqsubseteq p$ (3) $p \nearrow q$: the video segment according to p is before that according to q (4) $p \nwarrow q$: $p \nwarrow q$ iff $q \nearrow p$.*

Definition 3. *[Attribute Operators] Let $S \in \{\aleph, \Re, \Im\}$. And suppose that $v \in 2^S$, and $c \in S$.* **Attribute operators** *$\prec$, $\succ$, $\sim$ are defined as follow: (1) $S \neq \Im$: $v \prec c$ iff there exists an element $c' \in v$ such that $c' < c$. (2) $S \neq \Im$: $v \succ c$ iff there exists an element $c' \in v$ such that $c' > c$. (3) $v \sim c$ iff $c \in v$. I-operators are the special attribute operators which are applied on Ω values.*

Definition 4. *[Atomic Selection Condition] An* **atomic selection condition** *is defined as follows: (1) $a = b$ and $a \neq b$ are atomic selection conditions if a and b (variables or constants) are of the same type. (2) $a < b$, $a > b$, $a \leq b$ and $a \geq b$ are atomic seletion conditions if a and b are of the same type $\mathcal{T}$ and $\mathcal{T}$ has orderings $<$, $>$, $\leq$ and $\geq$ on its instances.*

Definition 5. *[Selection Condition] A* **selection condition** *is a boolean combination (i.e., an expression using the logical connectives $\neg$, $\vee$ and $\wedge$) of terms that have one of the following forms: (1) an* atomic-condition *(2) γ* **op** *$constant_1$ (3) $f_{REL}(\sigma_1)(\delta_1, \delta_2) = constant_2$ (4) $f_{REL}(\sigma_1)(\delta_1, \delta_2) = f_{REL}(\sigma_2)(\delta_3, \delta_4)$; where* atomic-condition *is an atomic selection condition, σ_i a video,* **op** *an attribute*

operator in $\{\sqsubseteq, \sqsupseteq, \nearrow, \nwarrow, \prec, \succ, \sim\}$, γ *a semantic attribute,* δ_i *an semantic object,* $constant_i$ *a constant value so that* $\{constant_1\} \in f_{DOM}(\gamma)$ *and* $constant_2 \in ALLEN$.

The two query languages are described below.

3.1 Video algebra

Queries in algebra are composed using a collection of operators. According to the different types of outputs, operators are presented as follows.

- **Boolean operator** (Syntax: $?(expr)$, Return: a Boolean value): *expr* is a selection condition. If the condition is true, the answer is YES. Otherwise, the answer is NO.
- **Physical operation** (Syntax: $\phi_{expr}(\sigma)$, Return: an Ω value): *expr* is a selection condition, and σ is a video. The returned is an Ω value ω so that *expr* is true during ω.
- **Semantic operation** (Syntax: $\psi_{expr}(\sigma)$, Return: a value in Δ): *expr* is a selection condition, and σ is a video. The returned is a semantic object δ that is true about *expr* in the context of video σ.
- **Projection** (Syntax: $\pi_{expr}(\sigma)$, Return: a Σ value): *expr* is a selection condition, and σ is a video. The returned is a new video σ' that is part of σ. Those segments that do not satisfy *expr* are cut off. Related time information is modified to be consistent with the context of the new video.
- **Composition** (Syntax: $\chi(\sigma_1, \sigma_2)$, Return: a Σ value): σ_1 and σ_2 are videos. The returned is a new video σ that includes σ_1 and σ_2.
- **Updates**: In the real-world it is likely that the database will change over time. Update operations are α (Insertion), θ (Deletion) and μ (Modification).
  - `Insertion` (Syntax: $\alpha_{expr,expr'}(\sigma)$, Return: a Σ value): σ is a video, *expr* is a selection condition and *expr'* is of the form $\gamma = constant$ where $\gamma \in f_\Gamma(\sigma)$, $\gamma \neq ID$ and $constant \in f_{DOM}(\sigma)(\gamma)$. First, all the semantic objects satifying *expr* are selected. If nothing is selected, a new semantic object equivalent to *expr* is inserted to the database.Otherwise, for each δ among them, if γ is an attribute of δ, its value will be changed to $\gamma(\delta) \sqcap constant$ if $\gamma = TIME$, $\gamma(\delta) \cup constant$ if $\gamma \neq TIME$. If γ is not an attribute of δ, it becomes an attribute with value *constant*.
  - `Deletion` (Syntax: $\theta_{expr,expr'}(\sigma)$, Return: a Σ value): σ is a video, *expr* is a selection condition and *expr'* is of the form $\gamma = constant$ where $\gamma \in f_\Gamma(\sigma)$and $constant \in f_{DOM}(\sigma)(\gamma)$. Firstly, all the semantic objects satifying *expr* are selected. For each δ among them, if γ is not an attribute of δ, nothing is changed. Otherwise, if $\gamma = ID$ and $\gamma(\delta) = constant$, δ is deleted from the database. On the other hand, its value will be changed to $\gamma(\delta) \setminus constant$ if $\gamma \neq TIME$, ω if $\gamma = TIME$, where $\omega \sqsubseteq \gamma(\delta)$ and ω do not overlap with the interval *constant*.
  - `Modification` (Syntax: $\mu_{expr,expr'}(\sigma)$, Return: a Σ value): σ is a video, *expr* is a selection condition, and *expr'* is of the form $\gamma = constant$

where $\gamma \in f_\Gamma(\sigma)$ and $constant \in f_{DOM}(\sigma)(\gamma)$. Firstly, all the semantic objects satifying *expr* are selected. For each δ among them, if γ is not an attribute of δ, nothing is changed. Otherwise, its value will be set to *constant*.

Examples of video algebra queries We now present several examples to illustrate how to write queries in video algebra. The database in the previous section is used for our examples. We will use parentheses as needed to make our algebra expressions unambiguous. Let σ represent the video. Let $\mathcal{A}$, $\mathcal{S}$, $\mathcal{C}$ and $\mathcal{T}$ denote attributes ACTION, SUBJECT, CONTAIN, and TIME respectively.

The user is interested in the scene where a dead body appears after somebody has assembled a rifle, and wants to know if it is true that the scene belongs to interval [12, 15], he or she can take the following steps: (1) $\delta_1 = \psi_{(\mathcal{A}\sim\text{“}die\text{”})\wedge(\mathcal{S}\sim\text{“}person\text{”})}(\sigma)$ (2) $\delta_2 = \psi_{(\mathcal{A}\sim\text{“}assemble\text{”})\wedge(\mathcal{S}\sim\text{“}rifle\text{”})\wedge(\mathcal{S}\sim\text{“}murderer\text{”})}(\sigma)$ (3) $p = \phi_{f_{REL}(\sigma)(\delta_1,\delta_2)=BEFORE}(\sigma)$ (4) $?(p \sqsubseteq [12, 15])$. The returned value of the last expression is the answer to the above query.

Now suppose that there is a collection of videos. The user might want to create a new video containing only segments of interest from the collection to be used later. For instance, the user wants a new video that is related to the murder only. He or she can write: (1) $\sigma_1' = \pi_{\mathcal{S}\sim\text{“}murderer\text{”}}(\sigma_1)$ (2) $\sigma_2' = \pi_{\mathcal{S}\sim\text{“}murderer\text{”}}(\sigma_2)$ (3) $\sigma' = \chi(\sigma_1', \sigma_2')$. Sometimes it is possible that the database contains incomplete information, so we may later want to modify or insert more knowledge. For instance, from some source of information extraction, we know that the scene in which the murderer smiles and his teeth are seen (semantic object δ_6) corresponds to period [6, 7], this information can be inserted to the database by the query: $\sigma = \alpha_{ID=6,\mathcal{T}=[6,7]}(\sigma)$

For some reason, semantic object δ_{12} does not exist anymore because of an earlier deletion. It is expected to take it out of semantic object δ_{11}. The query for this is $\sigma = \theta_{ID=11,\mathcal{C}=12}(\sigma)$

3.2 Video calculus

We introduce a video calculus, which can be considered an extension to the relational calculus, as an alternative to video algebra. It allows us to describe the set of answers without being explicit about how they should be computed. As in relational calculus, the language for writing *formulas* is the heart of our calculus. We now define these concepts formally, beginning with the notion of a formula.

Syntax of calculus queries

Definition 6. *[Atomic formula] Let Δ' be any set of semantic objects, σ_1, σ_2 video variables, ω_1 and ω_2 interval variables, δ_1 and δ_2 semantic object variables, and γ a semantic attribute. Let* **op** *denote an operator in the set $\{<, >, =, \leq, \geq, \neq, \sqsubseteq, \sqsupseteq, \nearrow, \nwarrow, \prec, \succ, \sim\}$. An* **atomic formula** *is one of the following: (1)*

$\delta_1 \in \Delta'$ *(2)* ω_1 **op** ω_2 *(3)* $\gamma(\delta_1)$ **op** $\gamma(\delta_2)$ *(4)* $\gamma(\delta_1)$ **op** *constant (5)* $TIME(\delta_1)$ **op** ω *(6)* $f_{REL}(\sigma)(\delta_1, \delta_2) = constant$ *(7)* $f_{\Delta}(\sigma_1) = constant$ *(8)* $f_{\Delta}(\sigma_1) = f_{\Delta}(\sigma_2)$.

Definition 7. *[Formula] A* **formula** *is recursively defined to be one of the following, where p and q are themselves formulas, and p(T) denotes a formula in which the variable T appears: (1) any atomic formula (2)* $\neg p$, $p \wedge q$, $p \vee q$, *or* $p \Rightarrow q$ *(3)* $\exists T(p(T))$, *where T is a variable and* $T \in \Sigma \cup \Omega \cup \Delta$ *(4)* $\forall T(p(T))$, *where T is a variable and* $T \in \Sigma \cup \Omega \cup \Delta$

In the last two clauses above, $\exists$ and $\forall$ are two quantifiers in traditional logic, and are said to *bind* the variable T.

Definition 8. *[Free variable] A variable is said to be* **free** *in a formula or a sub-formula (a formula contained in a larger formula) if the (sub-)formula does not contain an occurrence of a quantifier that binds it.*

And now is time for the formal definition of a video calculus query.

Definition 9. *[Calculus query] A* **calculus query** *is defined as an expression of the form* $\langle type-of-output \rangle \{T \mid p(T)\}$ *or the form* $\langle \gamma_1, \gamma_2, .., \gamma_n \rangle \langle type-of-output \rangle \{T \mid p(T)\}$ *where* γ_i *for* $i \in \{1, 2, .., n\}$ *is a semantic attribute, T is the only free variable in the formula p(T) and is of type* $type-of-output \in \{\Sigma, \Omega, \Delta\}$.

Semantics of calculus queries The answer to a calculus query $[\langle \gamma_1, \gamma_2, .., \gamma_n \rangle]$ $\langle$*type-of-output*$\rangle \{T \mid p(T)\}$, as we noted earlier, is the set of all values t of type *type-of-output* so that the formula $p(T)$ evaluates to TRUE with variable T assigned the value t. To complete this definition, we must state which value assignments to free variables in a formula make the formula TRUE.

A query is evaluated on a given instance of the video database. Let each free variable in a formula F be bound to a value. For the given assignments of values to variables, with respect to the given video database instance, the formula F is TRUE if one of the following holds:

- F is an atomic formula $\delta \in \Delta'$, and δ is a variable assigned a semantic object in the instance of Δ'.
- F is of an atomic formula $\omega \in \Omega$, and ω is a variable assigned an interval in the instance of Ω.
- F is an atomic formula $\gamma(\delta_1)$ **op** $\gamma(\delta_2)$, and the semantic objects assigned to δ_1 and δ_2 have attribute values $\gamma(\delta_1)$ and $\gamma(\delta_2)$ that make the comparison TRUE.
- F is an atomic formula $\gamma(\delta)$ **op** *constant*, and the semantic object assigned to δ has an attribute value $\gamma(\delta)$ equal to *constant*.
- F is an atomic formula $TIME(\delta)$ **op** ω, and the semantic object assigned to δ has an attribute value $TIME(\delta)$ that makes the comparison TRUE.
- F is an atomic formula $f_{REL}(\sigma)(\delta_1, \delta_2) = constant$, and the semantic objects assigned to δ_1 and δ_2 have value $f_{REL}(\sigma)(\delta_1, \delta_2)$ that makes the comparison TRUE.

- F is an atomic formula $f_{\Delta}(\sigma) = constant$, and the video assigned to σ has the set of semantic objects $f_{\Delta}(\sigma)$ equal to $constant$.
- F is an atomic formula $f_{\Delta}(\sigma_1) = f_{\Delta}(\sigma_2)$, and the videos assigned to σ_1 and σ_2 have the same set of semantic objects.
- F is of the form $\neg p$, and p is not TRUE; or of the form $p \wedge q$, and both p and q are TRUE; or of the form $p \vee q$, and one of them is TRUE; or of the form $p \Rightarrow q$, and q is TRUE whenever p is TRUE.
- F is of the form $\exists T(p(T))$, and there is some assignment of values to the free variables in $p(T)$, including T, that make it TRUE.
- F is of the form $\forall T(p(T))$, and there is some assignment of values to the free variables in $p(T)$ that make it TRUE no matter what value is assigned to variable T.

Examples of video calculus queries We now illustrate the video calculus through several examples. The video database example in section 2 is used for this purpose. The query "Show me the segments of video where the murderer appears with the gun" can be expressed by the calculus query: $\langle \Omega \rangle \{\omega \mid \forall \delta(\delta \in \Delta \wedge \mathcal{S}(\delta) \sim \text{“}murderer\text{”} \wedge \mathcal{S}(\delta) \sim \text{“}gun\text{”} \wedge \mathcal{T}(\delta) \sqsubseteq \omega\}$. If the user would like to find what the murderer does before committing murder while a man is placing money into a briefcase, the query below can be used:

$$\begin{aligned} &\langle \mathcal{S}, \mathcal{A} \rangle \{\Delta\} \{\delta \mid \delta \in \langle \mathcal{S}, \mathcal{A} \rangle \{\Delta\} \{\delta \mid \mathcal{S}(\delta) \sim \text{“}murderer\text{”} \\ &\wedge \exists \delta'(\delta' \in \Delta \wedge \mathcal{S}(\delta') \sim \text{“}person\text{”} \wedge \mathcal{A}(\delta') \sim \text{“}die\text{”} \wedge \mathcal{T}(\delta') \nwarrow \mathcal{T}(\delta)\} \\ &\wedge \exists \delta'(\delta' \in \Delta \wedge \mathcal{S}(\delta') \sim \text{“}person\text{”} \wedge \mathcal{S}(\delta') \sim \text{“}money\text{”} \\ &\wedge \mathcal{S}(\delta') \sim \text{“}briefcase\text{”} \wedge \mathcal{A}(\delta') \sim \text{“}place\text{”} \\ &\wedge f_{REL}(\sigma)(\delta, \delta') = DURING)\} \end{aligned} \tag{1}$$

The user might want a new video containing information about the murderer only, he or she can use the query: $\langle \Sigma \rangle \{\sigma \mid \forall \delta(\delta \in f_{\Delta}(\sigma) \wedge \mathcal{S}(\delta) \sim \text{“}murderer\text{”})\}$.

4 Concluding Remarks

In this paper, we introduced a semantic video data model called SemVideo with the following properties: (1) Some semantic contents not having related time information are modeled like ones that do; (2) Not only the temporal feature of semantic descriptions, but also the temporal relationships among themselves are components of the model.

An advantage of SemVideo model is that various types of knowledge captured by different semantics extraction techniques are utilized. In a long run, we expect that the metadata of a video database system will be built from various sources of information and continue being updated with more kinds of knowledge captured by more sources. The idea of SemVideo is on this track. Based on the model, we derived formalisms for the query system which provide reasonably powerful capabilities to end-users for their efficient questioning about the database. In

particular, we proposed two formal query languages that are a (procedural) video algebra and a (declarative) video calculus and include a comprehensive set of query formations working at both video and within-video level.

References

1. S. Adali, K. S. Candan, S.-S. Chen, K. Erol, and V. S. Subrahmanian. Advanced video information system: Data structures and query processing. *ACM-Springer Multimedia Systems Journal*, 1996.
2. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM, Springer-Verlag*, 26(11), 1983.
3. E. Ardizzone and M. L. Cascia. Automatic video database indexing and retrieval. *Multimedia Tools and Applications*, 4(1), 1997.
4. S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong. Videoq: An automated content based video search system using visual cues. In *ACM Multimedia*, November 1997.
5. T.-S. Chua and L.-Q. Ruan. A video retrieval and sequencing system. *ACM Transactions on Information Systems*, 13(4), 1995.
6. Y. Day, S. Dagtag, M. Iino, A. Khoakhar, and A. Ghafoor. A multi-level abstraction and modeling in video databases. *ACM Springer-Verlag Multimedia Systems*, 7(5), 1999.
7. C. Decleir and M.-S. Hacid. A database approach for modeling and querying video data. In *IEEE Data Engineering*, Australia, 1999.
8. A. Hampapur, R. Jain, and T. Weymouth. Production model based digital video segmentation. *Journal of Multimedia Tools and Applications*, 1(1), 1995.
9. H. Jiang, D. Montesi, and K. Elmagarmid. Videotext database systems. In *IEEE Int'l Conf. on Multimedia Computing and Systems*, Ontario, Canada, June 1997.
10. J. Oh and K. A. Hua. An efficient and cost-effective technique for browsing, querying and indexing large video databases. In *ACM SIGMOD*, Dallas, TX, May 2000.
11. J. Oh, K. A. Hua, and N. Liang. A content-based scene change detection and classification technique using background tracking. In *SPIE Conf. on Multimedia Computing and Networking*, San Jose, CA, January 2000.
12. E. Oomoto and K. Tanaka. Ovid: Design and implementation of a video-object database system. *IEEE Trans. on Knowledge and Data Engineering*, 5, August 1993.
13. Y. Rui, S. Huang, and S. Mehrotra. Constructing table-of-content for videos. *ACM Springer-Verlag Multimedia Systems*, 7(5), 1999.
14. T. G. A. Smith and G. Davenport. The stratification system: A design environment for random access video. In *Proceedings of the 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, 1992.
15. D. Swanberg, C.-F. Shu, and R. Jain. Knowledge guided parsing in video databases. In *SPIE Conf. on Image and Video Processing*, volume 1908, San Jose, CA, February 1993.
16. R. Weiss, A. Duda, and D. Gifford. Content-based access to algebraic video. In *IEEE Int'l Conf. on Multimedia Computing and Systems*, Boston, USA, 1994.
17. A. Yoshitaka, Y. Hosoda, M. Yoshimisu, M. Hirakawa, and T. Ichikawa. Violone: Video retrieval by motion example. *Visual Languages and Computing*, 7, 1996.

Toward a User-Centered Image Retrieval System

Iadh Ounis

University of Glasgow
Glasgow G12 8QQ
Scotland, UK
E-mail: ounis@dcs.gla.ac.uk

Abstract. Experimental evidence accumulated over the past years indicated the importance of the ranking process in information retrieval. As for the textual documents, image ranking is a task that involves different parameters. They depend on the intrinsic characteristics of an image, but also on the indexing language used for representing its semantic content. We developed a weighting model that combines these parameters in a general scheme. Finding the best balance between the parameters is not straight-forward. Different parameter combinations leads to different rankings, which may be more or less accepted by the users. In this paper, we choose a set of test queries and present the impact of the parameters on the rank of each image. Different combinations are discussed, and the best combination is specified. For the evaluation, we follow a user-oriented approach, and compare the ranking provided by each parameter combination to the ranking given by human judgment. This is a step toward a user-centered image retrieval system, which will dynamically adapt to the user's profile and preferences.

1 Introduction

Images constitute a complex type of media, where the identification of the objects appearing in the image and of the relationships between them is a user-dependent process. Indeed, image evaluation is often the result of a cognitive process, in which perception may constitute the prevalent criterion. The importance of an image, according to the perceptive user evaluation, is often related to intrinsic image parameters. Examples of such parameters are object area, object position (in the center or in the corner), color, contrast, clarity, presence of the objects in the foreground or in the background [1, 2]. In addition, in order to cope with the image complexity, rich formalisms are often used as indexing languages [3–5]. As the formalism gets more complex, its specificities become more important and should be taken into account in a weighting scheme. This is due to the fact that the matching function, which is launched when the user submits a query and retrieves the relevant images, implies specific treatments related to the formalism. For instance, the genericity/specificity relation between two concepts "man" and "human" can be used by the matching function, to retrieve an image showing a man when the query mentions its more generic concept human.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 51–64, 2000.

An image weighting model should take into account both the parameters depending on the image characteristics and the parameters depending on the formalism specificities. The question is to which degree each of the parameters influences the final ranking for each query, and which parameter combination can be considered to be the best, from the user's point of view. The answer to this question would allow to efficiently implement our previously developed weighting model, which captures all the parameters in an adjustable scheme [6]. Implementing all the parameters is not feasible in a first step. For instance, automatic decision on the clarity of a certain object in the image would require specific expertise in image analysis and processing. We decided to take into account a few parameters which we judged sufficient for a first validation. On the other hand, all the parameters related to the formalism that we identified in our model are taken into account in the implemented system.

With these parameters available in the system, our purpose is to find the balance between them leading to the best ranking, according to user assessments. The rest of this paper is organized as follows. In section 2, we specify our model for an image collection. This allows us to rank the images retrieved by our system called RELIEF, as we show in section 3. The ranking of the system is discussed in section 4, for different values of the weighting parameters. In section 5, a user-oriented experiment is presented. It allows to evaluate different parameter combinations, as shown in section 6. Thus the best combination is identified. Its rankings are discussed in sections 7 and 8, with respect to the judgment of the users. This allows to improve the implemented system. A conclusion and future work are given in section 9.

2 A Weighting Model Applied to an Image Test Collection

The parameters that we listed above are part of a general weighting scheme, that applies for multimedia documents indexed by rich formalisms [6]. The model captures two types of weighting parameters: parameters that depend on the image characteristics and parameters depending on the indexing formalism specificities. When the documents are images, parameters such as color, area and contrast are taken into account for weighting each image and thus determine its rank, for a given user's query. In this paper, we apply this model to an image test collection. This collection is a set of 650 images that answer to specialized needs related to history, archaeology and culture [7].

The collection is indexed using the conceptual graphs (CGs) formalism [4]. Therefore, the weighting parameters implemented in our system, depending on the indexing language, are given by the specificities of the CGs formalism. The images are thus each given a weight, which is mainly computed from the concept type lattice, which is part of the ontology of the application domain. For instance, in such a concept type lattice we can find knowledge like MAN $\leq$ HUMAN, to say that all men are humans.

An image indexed by a graph whose information content is greater is given a greater weight. This follows the fact that in CGs the matching function is built on the genericity/specificity relation between concepts, and we favor the more specific graphs. For instance, a graph [HUMAN] → (Position) → [SEATED] is given less weight than a graph [WRITER] ← (Profession) ← [MAN: Victor Hugo] → (Position) → [SEATED]. Indeed, the second graph specifies more the type of the concept, i.e. we know that in fact we have a man, which is more specific than human; it identifies him, as being Victor Hugo; and it gives his profession. The information content is greater for the second graph, and the image that is indexed by this graph will be ranked before an image indexed by the first graph, for a query looking for a seated human.

For the parameters dependent on the image characteristics, we consider in our implementation the area of the objects, the number of occurrences of objects in the same image, and the composition level. The latter parameter is extracted from a composition hierarchy that groups the objects identified in each image. An example would be an hierarchy that describes the fact that an image is composed of a building and a tree, and the building is in turn composed of windows, walls and doors. Going back to the image ranking, an object which has a larger area in the image is considered to be more important, and the image is given a larger weight and favored in the ranking. Similarly, images containing several occurrences of the same object are privileged as compared to those with less occurrences of the object. Finally, an image composed of a statue which is turn composed of a woman and a child is given less weight than an image directly composed of a woman and a child, if the user is looking for a woman. Indeed, the composition level parameter is greater if the object is closer to the top of the hierarchy. The application of our weighting scheme to the parameters mentioned here gives the following ranking formula:

$$\begin{aligned} \text{RELEVANCEVALUE}(d,q) = b \times W_{\text{IMAGE}} + \\ (1-b) \times W_{\text{CONCEPTUALGRAPHS}} \end{aligned} \tag{1}$$

where $\text{W}_{\text{IMAGE}} = \sum_{i=1}^{n} (a_{\text{AREA}} \times \text{AREA} + a_{\text{COMPLEVEL}} \times \text{COMPLEVEL})$; n is the number of occurrences, and it denotes the number of objects in the image that correspond to the objects mentioned in the query.

The parameters AREA and COMPLEVEL are related to the characteristics of the image. Their impact on the image dependent weight W_{IMAGE} is controlled by a_{AREA} and $a_{\text{COMPLEVEL}}$ respectively. The weight related to the conceptual graph index is given as mentioned above. The relevance value of a document d for a query q is then computed as a weighted sum of the two types of weights. The parameter b is used to adjust the impact of the image characteristics and of the formalism specificities.

In the following, we are looking for the best combination of these parameters, i.e. b, a_{AREA} and $a_{\text{COMPLEVEL}}$, in the case of our image test collection and with respect to the user's preferences.

3 Introducing Ranking in RELIEF

We have developed an image retrieval system based on the conceptual graph formalism, and called RELIEF [4]. The first version of the system only classified images in several relevance classes, without any further ranking within a class. We introduced our weighting model into this system, which allows for having a totally ordered list of retrieved images [6]. For each displayed image, the relevance value computed with the formula 1 is shown in the form of a percentage, as shown in figure 1. In this case, the submitted query is "man on the right of a table". This ranking is obtained for the following parameter values: $b = 0.85$, $a_{\text{AREA}} = 0.5$ and $a_{\text{COMPLEVEL}} = 0.5$. This means that the image dependent parameters are favored as compared to those depending on the formalism. Such a choice follows the observation that image evaluation highly depends on the user's perception. The characteristics of the image are given equal importance. As for the number of occurrences of the man and of the table in each image, it is automatically introduced as part of the tests performed by the matching function.

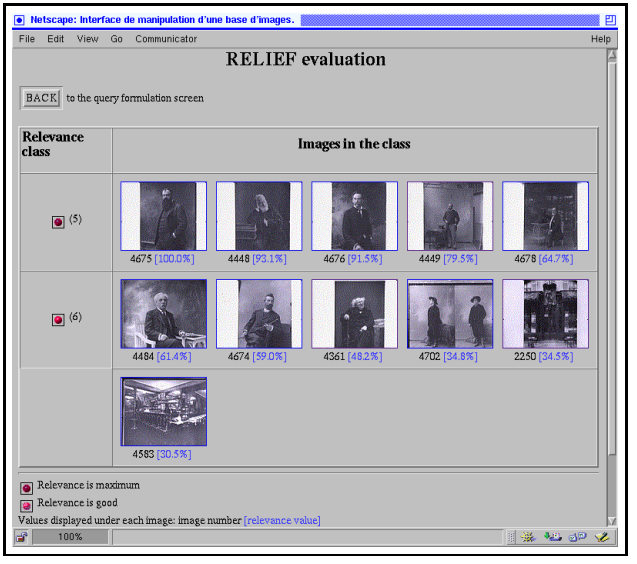


Figure 1. The images retrieved for the query "man on the right of a table" are ordered according to our model.

As one can see, the five images of the first relevance class are mainly ordered according to the area of the objects man and table. This is due to the fact that the composition level does not play any role here, as the man and the table are always direct children of the top of the image composition hierarchies for our five images. It would not be the case in an image which is not a portrait, but a painting displaying a man and a table. In the following, we analyze the impact of different parameter values on the image ranking.

4 Image Ranking for Different Parameter Values

All along this section, we focus our discussion on queries for which the number of retrieved images is smaller, and therefore more easy to analyze. For the same parameter values as those used in the previous section, the images retrieved for the query "beard and hat" are ordered as shown in figure 2. As one can see, the area is still the prevalent criterion in the image ranking.

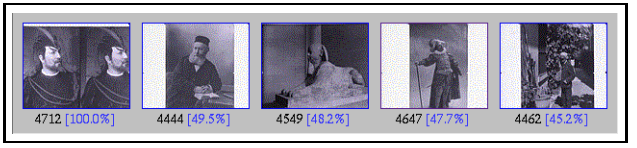


Figure 2. The ranked images for the query "beard and hat" for $b = 0.85$, $a_{\text{AREA}} = 0.5$ and $a_{\text{COMPLEVEL}} = 0.5$

Changing the parameter values greatly influences the relevance value of each image and the ranking changes accordingly. In a first step, we emphasize the formalism specificities, by taking b = 0.05 in formula 1. We keep the same values for image characteristics. The order is shown in figure 3. A first remark is that the relevance values are more compressed; the last value increases from 45.2% to 80.5%. The explanation of this fact is related to the graphs that index our images. As we mentioned in [4], the size of the graphs is almost constant in our collection, and thus the difference between the values is mainly given not by the size, but by the specificity of the concepts used in the indexes. Moreover, as the depth of the concept type lattice used in our collection is relatively low, the relative difference between specificities of concepts is low. We note that the graphs give close values to the retrieved images.

Figure 3. The ranked images for the query "beard and hat" for $b = 0.05$, $a_{\text{AREA}} = 0.5$ and $a_{\text{COMPLEVEL}} = 0.5$

In the ranking in figure 3, the image 4647 is given more importance because it is indexed by more specific concepts. For instance, the identified image object man is indexed not by man, but by the more specific concept composer, while the index of the last image 4444 introduces simply a man. In addition, the number of identified objects in the image 4647 is greater, and each one introduces a concept in the index. Thus the information content is the main criterion in the ranking, following the judgment of the specialists of the French Ministry of Culture who provided the original indexes of our images and who considered that, in particular, image 4647 contains more objects of interest than image 4444. Following our model, the formalism specificities are to be combined with image characteristics. While the formalism specificities are somewhat hidden to

the user and, in addition, they depend on the choices made during the indexing process, the perceptive factors have a decisive influence on an image evaluation.

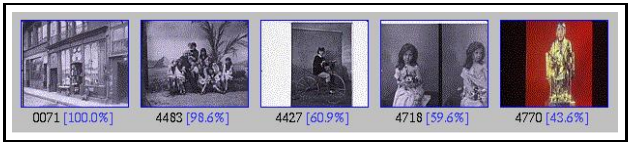


Figure 4. The ranked images for the query "seated child" for $b = 0.7$, $a_{\text{AREA}} = 0.2$ and $a_{\text{COMPLEVEL}} = 0.8$

Figure 5. The ranked images for the query "seated child" for $b = 0.7$, $a_{\text{AREA}} = 0.5$ and $a_{\text{COMPLEVEL}} = 0.5$

Figure 6. The ranked images for the query "seated child" for $b = 0.7$, $a_{\text{AREA}} = 0.8$ and $a_{\text{COMPLEVEL}} = 0.2$

Let us change the query, and take "seated child" instead. For the same parameter $b = 0.7$, we obtained the results in figures 4, 5 and 6 for different impacts of the area and composition level parameters. It is interesting to see how image 4718 becomes more important, from the fourth position when the area parameter is minimum to the top when area has maximum impact. The first position of image 0071 in figure 4 is in fact given by the formalism. Indeed, the area is given low importance (0.2), composition level is emphasized (0.8) but it remains almost the same among the images, so the graph specificities make the difference. As we increase the importance of the area parameter, the image 0071 goes down in the ranking, according to its relevance value that varies from 100% to 87.2%. The reader may note the constant position of image 4770. The area of the child identified in the image is small (there is a woman with a seated child in that image) and the composition level parameter also plays his role (the woman and the child are part of a statue). Finally, the impact of the number of occurrences of the object child in the relevance value is illustrated by image 4483. There is a group of children, and each occurrence adds a certain value to W_{IMAGE} in the formula in section 2. The relevance values of images 4483 and 0071 are almost the same in the three combinations, as the richness of the index graph of image 0071 compensates for the numerous occurrences of the object child in image 4483. We note that in the last combination in figure 6, image 4718 is placed before image 4483 because the area of the identified girl is greater than the sum of the areas of the children identified in image 4483.

The examples taken in this section illustrate the importance of the choice of a certain combination of parameters in the final system ranking. In order to have an overall view and to find the values that match the best with the preferences of the users, we ran a set of 7 queries for different combinations of the parameters:

Number	Query	Number of retrieved images
q_1	man on the right of a table	11
q_2	horse and vehicle	12
q_3	beard and hat	5
q_4	cobblestone street and tree	7
q_5	seated child	5
q_6	sea and boat	7
q_7	automobile	8

Each combination of parameters defines a particular system, that we note $\mathcal{S}_i$ $\langle$ b, a_{AREA}, $a_{\text{COMPLEVEL}}$ $\rangle$. We choose to introduce 10 systems. They may be considered to be representative for a class of parameter combinations. Each query launched within a system gives some particular relevance values, hence a particular ranking.

5 An Experiment with Users

In order to extract the interesting features of the systems $\mathcal{S}_i$, we choose a user-oriented approach, and compare the system rankings to the user rankings. This allows to emphasize, among the 10 systems, which one matches the best with the user's preferences. The best parameter combination corresponds to that system. In our experiments, we asked 6 users to assess the ranking of the sets of relevant images retrieved by the system, given in arbitrary order for each of the queries listed above. The users were given the images as they were retrieved, i.e. in the same size. The user group is composed of two women and four men, all of them young (aged between 24 and 35). We have an associate professor and five computer science graduate students. Three of the users wear glasses, which is an important detail for an experiment involving the evaluation of rich documents like images. Without exception but more definitely for the users wearing glasses, the small size of the given images seems to be an inconvenient. In the user group, there is an user who knows well the image collection and the previous version of the system, without any knowledge about our weighting model. The rest of the users are not familiar with the system, nor with the collection of images.

The evaluation of the seven queries and the ordering of their results took between 5 and 15 minutes for each user. A general remark is that, while for the first images it is relatively easy for the user to choose his or her best images, ordering the rest of the images implies a cognitive overload, clearly shown by the users. The ranking of queries whose answers are numerous (more than 10 images) takes comparatively much more time than for the other queries. In most cases, ranking the images retrieved for the queries q_1 and q_2 took about the half of the time dedicated to the experiment. A practical consequence is that, in the following, we pay more attention to the first 5 images in the complete ranking provided by the users. We consider that for these images the user's choice is

adequate, which is not the case for the rest of the images. In fact, some of the users explicitly mentioned that the images they put at the end of their rankings are all of about the same importance to them, and acknowledged the fact that the ranking they give is not decisive in this case. In table 1, we present the first 5 images indicated by the users for our queries. The image numbers are given in each column, in vertical order.

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
	4448	4593	4712	1124	4728	4600	4493
	4449	4598	4444	1129	4483	4562	4389
User 1	4678	4599	4647	0142	0071	4564	4630
	4676	1355	4462	1128	4427	4741	2786
	4675	0305	4549	2707	4770	4606	1496
	4449	4593	4444	0142	4718	4741	4630
	4448	4599	4712	1124	4770	4606	4493
User 2	4675	4598	4647	1129	0071	4561	4389
	4676	1355	4462	2707	4483	2495	1496
	4678	0305	4549	1128	4427	4600	1271
	4449	4593	4444	1124	4718	4600	4630
	4448	4599	4712	0142	4427	4606	4483
User 3	4678	4598	4647	1129	4483	4741	4389
	4675	3794	4462	1280	4770	4562	1496
	4676	4767	4549	2707	0071	4561	4739
	4448	4599	4444	1124	4718	4606	4493
	4678	4593	4712	0142	4483	4562	4389
User 4	4449	4598	4647	1128	4427	4564	4630
	4675	3794	4463	1129	4770	4561	4554
	4676	1057	4549	1280	0071	4600	1496
	4678	4593	4712	1129	4718	4606	4493
	4449	4599	4444	1124	4770	4741	1496
User 5	4448	0305	4647	0142	0071	4561	4389
	4675	1355	4462	1128	4483	4564	4630
	4676	1057	4549	2707	4427	2495	1271
	4448	4593	4444	1124	4483	4562	4389
	4678	4599	4712	1280	4758	4741	4630
User 6	4449	4598	4647	0142	4427	4564	1496
	4675	0305	4462	0355	4970	4606	4493
	4676	3794	4549	2707	0071	4600	1271

Table 1. The ranking chosen by the 6 users

We note that for certain queries the choices of the users are almost identical (see the rankings given for q_3 or even for q_1). In order to combine the ranking of the different users in one overall reference ranking, we processed the data in table 1 on a statistical basis. To this effect, we ordered the images according to their overall score. For each image retrieved for a query, we compute its mean position in the overall ranking, from the 6 positions chosen by the users. For instance, the score for image 4448 in query q_1 is (1 + 2 + 2 + 1 + 3 + 1) / 6 = 10 / 6. For the queries with more than 5 answers, the images chosen by the users generally differ. In this case, all the images mentioned by the users in their rankings are combined, their score computed, and then the images with the 5 best scores are retained in the overall ranking. In case of equal scores, the image chosen more times in a good position is favored. This overall ranking for the 7 queries is given in table 2. It corresponds to a virtual user who combines all the preferences of the 6 users, and serves as reference in the evaluation of the different systems $\mathcal{S}_i$ $\langle$ b, a_{AREA}, $a_{\text{COMPLEVEL}}$ $\rangle$. In other words, the ranking provided by the parameter combinations should be as close as possible to that captured by the combined user ranking.

q_1			q_2			q_3		
Img.	Score	Rank	Img.	Score	Rank	Img.	Score	Rank
4448	10/6	1	4593	7/6	1	4444	8/6	1
4449	12/6	2	4599	12/6	2	4712	10/6	2
4678	16/6	3	4598	21/6	3	4647	18/6	3
4675	24/6	4	1057	32/6	4	4462	24/6	4
4676	28/6	5	1355	35/6	5	4549	30/6	5
q_4			q_5			q_6		
Img.	Score	Rank	Img.	Score	Rank	Img.	Score	Rank
1124	8/6	1	4718	7/6	1	4606	15/6	1
0142	14/6	2	4483	16/6	2	4741	18/6	2
1129	19/6	3	4770	21/6	3	4562	21/6	3
1280	29/6	4	4427	22/6	4	4600	24/6	4
1128	29/6	5	0071	24/6	5	4564	26/6	5
q_7								
Image	Score	Rank						
4493	11/6	1						
4630	14/6	2						
4389	14/6	3						
1496	23/6	4						
1271	35/6	5						

Table 2. Combined user ranking for our set of 7 queries

6 Evaluation of the Ranking Systems

The user experiment identifies the best possible ranking that the parameter combinations should provide, in the form of the combined user ranking, denoted by μ. As a comparison between two different rankings, we use the statistical deviation. The best system among the 10 parameter combinations is the one which gives the least deviation value, when the combined user ranking is used as reference. We take each of the 10 systems $\mathcal{S}_i$ $\langle$ b, a_{AREA}, $a_{\text{COMPLEVEL}}$ $\rangle$ and compute its deviation [1] from the user ranking by using the formula 2:

$$\sigma = \sqrt{\frac{(\mathcal{S}_i - \mu)^2}{N}} \tag{2}$$

The deviation is obtained by taking each of the N images in the combined ranking given by the users, and finding its rank given by the system $\mathcal{S}_i$. We have 5 images, so $N = 5$. In the following table, we give the deviation values for the 10 systems and for the 7 queries previously introduced.

System	q_1	q_2	q_3	q_4	q_5	q_6	q_7
$\mathcal{S}_1$	2.23	9.27	2.44	2.82	2.6	3.28	3.49
$\mathcal{S}_2$	2.09	1.89	1.26	3.28	0.63	3.13	2.09
$\mathcal{S}_3$	2.09	3.28	1.26	3.22	1.09	3.13	2.72
$\mathcal{S}_4$	2.36	5.70	2.09	3.19	2.44	3.54	3.22
$\mathcal{S}_5$	2.44	3.40	1.78	3.46	1.26	3.13	2.79
$\mathcal{S}_6$	2.79	4.69	1.78	3.28	1.89	3.13	2.79
$\mathcal{S}_7$	2.19	5.96	2.09	3.63	2.44	3.03	2.79
$\mathcal{S}_8$	2.09	2.75	1.26	3.28	1.09	3.13	2.75
$\mathcal{S}_9$	2.09	3.52	1.26	3.19	1.26	3.13	2.79
$\mathcal{S}_{10}$	2.09	4.47	0.63	3.31	1.89	3.13	2.72

So that to identify the best parameter combination, we order the systems according to their deviation. To this effect, we take each query and give the

[1] Deviation is the square root of the statistical variance.

first systems that have the least deviation. We only take the first 5 because the others do not influence the search of the best combination. We obtain the following table:

Query	Best systems
q_1	$\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_8$, $\mathcal{S}_9$, $\mathcal{S}_{10}$
q_2	$\mathcal{S}_2$, $\mathcal{S}_8$, $\mathcal{S}_3$, $\mathcal{S}_5$, $\mathcal{S}_9$
q_3	$\mathcal{S}_{10}$, $\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_8$, $\mathcal{S}_9$
q_4	$\mathcal{S}_1$, $\mathcal{S}_4$, $\mathcal{S}_9$, $\mathcal{S}_3$, $\mathcal{S}_2$
q_5	$\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_8$, $\mathcal{S}_5$, $\mathcal{S}_9$
q_6	$\mathcal{S}_7$, $\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_5$, $\mathcal{S}_6$
q_7	$\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_8$, $\mathcal{S}_5$, $\mathcal{S}_6$

From this table, it is clear that the combination associated to the system $\mathcal{S}_2$ $\langle$ 0.95, 0.95, 0.05 $\rangle$ gives the best results for our image collection. The only query which does not conform to this choice is q_4, in which case $\mathcal{S}_2$ is classified in the fifth position. An analysis of the results of system $\mathcal{S}_2$ allows to identify how its ranking can be improved, as we show in the next section.

7 How to Improve the Best System Combination ?

The differences between the ranking provided by the best system $\mathcal{S}_2$ and the combined user ranking generally confirm the remarks that we give in section 4. In the following, we analyze the differences and review their explanations. This enables us to conjecture future corrections in our implementation that would give results closer to the combined user ranking. All along this section, we use the table 2.

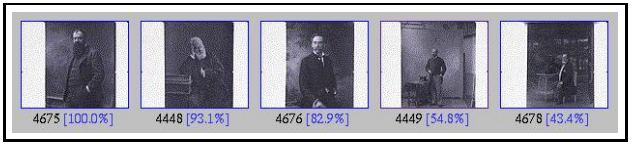


Figure 7. Ranked images for the query "man on the right of a table", for $\mathcal{S}_2$

For the first query, if we compare the results of $\mathcal{S}_2$ to the user combined ranking, we find that clarity is a parameter which plays an important role in the judgment of the users, at least for this collection. This observation confirms our weighting model. Indeed, we mentioned that images where the objects mentioned in the query are obvious should be given more importance, from the point of view of parameters depending on the image [6]. When the query mentions two objects, users seem to prefer images in which the two objects are clear, though possibly smaller, as in image 4449 for the query "man on the right of a table", while the images such as 4675, in which one object is very large and the other (the table) does not appear clearly (see figure 7), are considered to be less important by the users.

Figure 8. Ranked images for the query "horse and vehicle", for $\mathcal{S}_2$

The same remark is valid for the query q_2 (see figure 8), in which the main difference between the system ranking and the user ranking is given by the position of image 4767. The clarity parameter would solve the problem and, in the ranking provided by the system, the image 4767 would find its position as assessed by the users.

As for query q_1, the clarity parameter, once implemented, would adjust the system ranking towards the user's preference in the case of q_3.

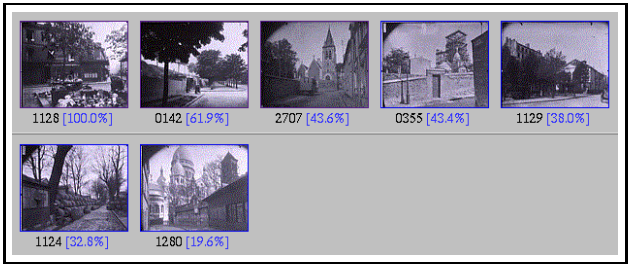


Figure 9. Ranked images for the query "cobblestone street and tree", for $\mathcal{S}_2$

For the query q_4 (see figure 9), the performance of the system $\mathcal{S}_2$ is not as close to the user's expectations as it is the case for the other queries. The main reason is the approximative indexing, which fails to identify the complete (and exact) contours of the trees occurring in the images (this explains the position of image 0142 in the system ranking). A more interesting issue is related to image 0355, which is constantly chosen at the end of the ranking by the users. We think that the reason is twofold: firstly, the street is not quite clear in this image; and secondly, and in our opinion more interestingly, the user background knowledge play their role in establishing the order between, for instance, images 1280 and 0355 (see table 2). Even though the street is not clear in both these images, the buildings on the right side of the street in image 1280, combined with the user knowledge that associate such a building alignment to the presence of a street, make it more easily to identify the street in image 1280 and thus to choose it in the first place among the two. The practical impact on the system and its

future extension is not obvious in this case, because managing such background knowledge is not straight-forward.

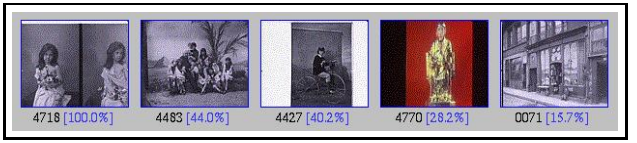


Figure 10. Ranked images for the query "seated child", for $\mathcal{S}_2$

For the query q_5 (see figure 10) the system matches very well the user preference. The position of image 4770 is in fact an advantage on the side of the system. Due to the size of the images, the users thought in the beginning that the statue itself was a child, and favored it in the ranking. However, it is not the case, and the child is in fact only a relatively small, unclear part of the image.

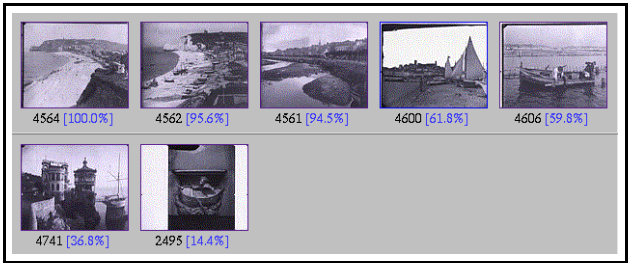


Figure 11. Ranked images for the query "sea and boat", for $\mathcal{S}_2$

The query q_6 (see figure 11) is another illustration of the cases when the users may be wrong in their decision, due to inexistent or incomplete knowledge on the context of the images. The system correctly puts the image 4606 in the sixth place, as the water that appears in the image is not part of a sea, but of an oyster bed. Therefore, it is not the water that corresponds to the "sea" formulated in the query, but the shore which appears in the background and is a more specific concept than sea in our concept type lattice. Perception here may be misleading, and this explains why the users globally choose it in the first place. Once the users understand that the image does not show a sea, they seem to change their ranking and follow the system's suggestion. The user's choice of image 4741 seems to be motivated by the clarity parameter, so introducing it in the system would improve its ranking with respect to the combined user reference ranking. The matching between the positions of image 4562 in the ranking provided by the system and in the user's preference confirm that the number of occurrences of an object is important when the user's background knowledge is involved. This applies at least for the boats that appear clearly (good contrast against the sand) in image 4562.

The last query q_7 further confirms that the clarity parameter would help the system to better match its ranking to what the users seem to prefer. Hence, from the above discussion, clarity seems to be an important parameter, that should be taken into account not only in our model, but also in its implementation on

RELIEF, next to the area and composition level parameters. This will improve in an important degree the ranking performance of the system.

8 Matching the Best System and the User Preferences

In this section, we show how far the rankings of the system $\mathcal{S}_2$ are from the rankings chosen by the users. We considered in our experiments 6 user assessments. Now, the ranking given by the system $\mathcal{S}_2$ may be considered to be an assessment by itself, so it would be interesting to compare it to the rankings given by each user. A detailed view is given in table 3, in the form of the deviation values, computed with the combined user ranking as reference.

Q	User 1	User 2	User 3	User 4	User 5	User 6	$\mathcal{S}_2$
q_1	0.63	1.26	0.63	0.63	1.26	0.63	2.09
q_2	1.48	0.89	1.34	2.32	2.19	0.63	1.89
q_3	0.63	0	0	0	0.63	0	1.89
q_4	1.18	1.09	0.44	1.09	1.48	1.89	3.28
q_5	1.26	1.41	1.09	0.63	1.41	0.89	0.63
q_6	2.6	1.78	1.61	2.09	1.94	1.89	3.13
q_7	0.89	0.63	1.09	1.18	1.26	1.67	2.09

Table 3. The deviation for the six users and for the system $\mathcal{S}_2$. The reference used in the computation is the combined user ranking.

In this table, the maximum value indicates a bigger deviation between the ranking of an user or the ranking of the system, and the combined user ranking. On the other side, a value of 0 is an indication that the corresponding user is the reference user, for that query. If the system deviation is lower than the maximum user deviation, we consider that the system is very good, as its assessment is comparable to the human judgment. If the value is comparable to the maximum user deviation, then the system is in our opinion good enough. Bigger differences may be an evidence of a hidden, non-included parameter or of an indexing problem. It is these values which formally allowed us to detect the problems and extract useful data for future development of the implemented system. Generally, the purpose is to have low deviation values. At a first glance, two conclusions can be drawn. Firstly, the assessment of the users are virtually identical for the query q_3, which is interesting for a rich image collection. Secondly, the system performs well, and it is particularly good for the query q_5. Indeed, there are several users whose ranking assessments deviate more from the combined user ranking than the system's ranking does. Globally, the system performance can be further improved if the indexing is more accurate and if parameters such as clarity of objects are implemented.

9 Conclusion and Future Work

The experiments performed and the user-oriented evaluation lead to interesting conclusions, which will allow us to improve the ranking of our system RELIEF. The results are very encouraging, for a first implementation. This is further supported by the richness and the degree of specialization of our collection, as

well as the perceptive factors related to image evaluation. The ranking works the best for a parameter combination which favors the perception factors.

Generally, the best parameter combination will depend on the document collection and on the user preference. A user study such as the one carried here only serves to evaluate the global performance of the system, and for future improvement; for efficiency reasons, it cannot be performed very often and for all the collections. An intelligent information retrieval system should be able to adapt to the working environment, i.e. to the document collection and to the user profile. In order to find the best parameter combination for a given user, the system must dynamically update the parameter values, starting from a certain default combination. Relevance feedback is a technique already used in information retrieval, to answer to assessments made by users during interactive sessions [8]. An application of this technique to our case would involve the search in an n-dimensional space, where n is the number of parameters. Given the current parameter value combination and a user feedback, that classifies a certain image before another one, the system must advance in the multidimensional space, towards a convergence point that represents the best combination. The next development system will focus on such aspects. Our final purpose is to build a user-centered image retrieval system.

References

1. J. May and P. Barnard. Modelling multimodal interaction: A theory-based technique for design analysis and support. In *Human-Computer Interaction (INTERACT '97)*, pages 667–668, London. Also in www.shef.ac.uk/ pc1jm/guide.html, 1997. Chapman & Hall.
2. A. Plante, S. Tanaka, and S. Inoue. Evaluating the location of hot spots in interactive scenes using the 3R toolbox. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'98)*, pages 117–123, Los Angeles, USA, 1998. ACM Press.
3. M. Mechkour. EMIR2. An extended model for image representation and retrieval. In *DEXA'95. Database and EXpert system Applications, London.*, pages 395–404, September 1995.
4. I. Ounis and M. Pasca. RELIEF: Combining expressiveness and rapidity into a single system. In *Proccedings of the 21th ACM SIGIR Conference, Melbourne, Australia*, pages 266–274, 1998.
5. I. Ounis and M. Pasca. Modeling, indexing and retrieving images using conceptual graphs. In *Proceedings of the 9th DEXA International Conference on Database and EXpert Systems Applications*, pages 226–239, Vienna, Austria, August 1998.
6. I. Ounis. A flexible weighting scheme for multimedia documents. In *Proceedings of the 10th DEXA International Conference on Database and EXpert Systems Applications*, pages 392–405, Florence, Italy, August 1999.
7. Y. Chiaramella and M. Mechkour. Indexing an image test collection. Technical report, FERMI BRA 8134, April 1997.
8. J.J. Rocchio Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART retrieval System–Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

MEASURING FOR DATABASE PROGRAMS MAINTAINABILITY

Mario Piattini[1] , Antonio Martinez[1,2]
[1]RESEARCH GROUP ALARCOS
DEPARTAMENTO DE INFORMÁTICA
UNIVERSITY OF CASTILLA-LA MANCHA
RONDA DE CALATRAVA, 5
13071, CIUDAD REAL. SPAIN.
{mpiattin, amartinez} @inf-cr.uclm.es
[2]EXCMA. DIPUTACION DE CIUDAD REAL
CALLE TOLEDO, 17
13001, CIUDAD REAL (SPAIN)
amartinez@dipucr.es

Abstract. Nowadays, relational databases are introduced in most of the Information Systems, becoming their essential core. The relational database language, SQL, is being increasingly used for application development. Software engineers have been putting forward huge quantities of measures for software products, processes and resources. Unfortunately, almost all the measures proposed until now, have focused on 3GL program characteristics disregarding databases and their associated languages. Only a few author have proposed some metrics for 4GL effort estimation. In this article we describe three simple measures for assessing SQL code maintainability. Both an experiment with students and a real case in a state owned organization have empirically validated these measures as maintainability indicators.

1 Introduction

Nowadays, relational databases are introduced in most of the Information Systems, becoming their essential core. The relational database language SQL[1], is used in almost all the organizations which have management information systems, either embedded in 3GL (Third Generation Language), mainly COBOL, programs or in more modern productivity-enhancing tools such as 4GL (Fourth Generation Languages).

Information Technology (IT) organizations must improve the quality of software products, especially maintainability, because maintenance cost are the most important problem of software development, ranging between 60 and 90 percent of life-cycle costs [2],[3]. Software measurement is widely recognized as an effective means to

[1] ISO/IEC and ANSI standarizarion committes have recently proposed a new version of the SQL know as SQL:1999 (formerly SQL3) [1]. This paper focus on the previous versions (SQL'89).

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 65–78, 2000.

understand, monitor, control, predict and improve software development and maintenance projects [4]. Software engineers have been putting forward hundreds quantities of measures for software products, processes and resources [5],[6]. Unfortunately, almost all the measures proposed until now, have focused on 3GL program characteristics disregarding databases and their associated languages [7], or object oriented environment [8]. Only some works attempt to estimate the effort of developing 4GL programs [9],[10],[11] but we have not came accross any efforts to try to assess SQL code maintainability. We adopted the definition of maintainability as the capability of the software product to be modified. Modifications may include corrections, improvements or adaption of the software to changes in environment, and in requirememnts and functional specifications [12]. Maintainability is achieved by three factors: understandability, modifiability and testability [13].
In this paper, three simple measures for assessing SQL program maintainability are proposed. In next section we describe the three measures. In section 3, an empirical validation experiment is presented. In section 4 the results of a real case study are exposed. Finally, in section 5 we summarize the paper and present the conclusions.

2 Proposed measures for assessing SQL code

SQL language is composed of different kinds of statements: definition statements (e.g. CREATE TABLE), manipulation statements (e.g. INSERT) and control statements (e.g. COMMIT). Four different manipulation statement can be found: SELECT, INSERT, DELETE and UPDATE. The first one is the most used and so we focus our work on it.
We propose the following three measures for characterizing SELECT statement:

NT measure

Number of tables referred in the SELECT statement.

NN measure

Number of nesting, considers the number of "SELECT" in the SELECT statement.

G measure

This measure indicates whether exist (1) or not (0) a GROUP BY clause in the SELECT statement.

```
   select f.name_emp, p.number_fic, p.date
        from control_employee p, employee f, h_employee h
        where p.id_emp not in
(select h.id_emp
           from control_employee p, employee f, h_employee h
             where p.number_fic=h.number_fic
                    and f.id_emp=h.id_emp
                    and p.id_emp=f.id_emp
                    and p.date='171298'
                    and p.control='SM'
                    and p.status='A'
                    and f.sex='V'
                    and p.hour in
(select hour
from control_employee p, employee f, h_employee h
                         where p.number_fic=h.number_fic
                                 and f.id_emp=h.id_emp
                                 and p.id_emp=f.id_emp
                                 and p.date='171298'
                                 and p.control='SM'
                                 and p.tipe='A0'
                                 and h.remaindert=0
                         )
             )
               and p.date='151298'
               and p.number_fic=h.number_fic
               and p.id_emp=f.id_emp
               and f.id_emp=h.id_emp
group by f.name_emp, p.number_fic, p.date
```

Fig. 1. In the example the values are NT=3, NN=3 and G=1.

These measures were proposed based on intuition and experience with SQL programs development and maintenance. The number of tables is likely to influence all the three maintainability factors as SQL statements will be more difficult to understand, to modify and to test if they include more tables.

The number of nesting is likely to also influence the maintainability of the SQL code, as each nesting demands a new level of thinking similar to a new call in 3GL or a level of inheritance in object-oriented programs [14]. The earlier relational optimizers had also been influenced by the SELECT nesting, and vendors recommended not to nest beyond 3 levels for performance reasons.

We believe that grouping rows for calculating values also influences maintainability of SQL programs as it implies an additional operation which must be carried out over a set of rows.

3 Empirical validation of the proposed measures

In this section we summarize an experiment done in order to validate NT, NN and G measures. This empirical validation has been carried out following the experimental method applied to software engineering [9],[15].

Our aim is to demonstrate that the proposed measures can be used for measuring the understandability of the SQL statement which influences its maintainability.

Understandability is the the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use [12].

3.1 Hypotheses

The formal hypotheses are:
Null hypothesis: Different values of the three measures do not affect the understandability of the SELECT statement.
Alternative hypothesis 1: The value of the NT measure affects the understandability of the SELECT statement.
Alternative hypothesis 2: The value of the NN measure affects the understandability of the SELECT statement.
Alternative hypothesis 3: The value of the G measure affects the understandability of the SELECT statement.
Alternative hypothesis 4: The combination of NT and NN measures affects the understandability of the SELECT statement.
Alternative hypothesis 5: The combination of NT and G measures affects the understandability of the SELECT statement.
Alternative hypothesis 6: The combination of NN and G measures affects the understandability of the SELECT statement.
Alternative hypothesis 7: The combination of NT, NN and G measures affects the understandability of the SELECT statement.

3.2 Subjects

The participants of the field test were Computer Science students at the University of Castilla-La Mancha (Spain), who were enrolled in a database course lasting two semesters. Until the day of the experiment, the students did not known that they were to do it. The experiment was developed by 34 students, but only 19 of them were selected, because it answered rightly all the questions.
We have tried to minimize variability among participants by choosing people of the same degree, in particular from the third year.

3.3 Experimental materials

Eight separate SQL statements were required to test the hypotheses. In each one values of the three measures were different. There were two possible values for NT (one or three), two for NN (one or three) and two for G (zero or one). The documentation accompanying each design was approximately twelve pages long including the tables and the queries. In appendix A the tables and the SQL statements are shown. The same table was used to reduce semantic variability in all the eight cases. In order to avoid learning errors, the eight cases were in different order for each subject, and the subjects were forced to follow the order in which cases appeared in the experimental material.

The subjects were asked to write down the initial and the final time, and the result for each query. We only counted the time employed in giving correct answers.

3.4 Experimental design.

Each level of one factor appears with each level of the other one, so we have selected the crossing design NT x NN x G. See table 1.

		FACTOR NT			
		LOW		HIGH	
		FACTOR NN			
		LOW	HIGH	LOW	HIGH
FACTOR G	LOW	1,1,0	1,3,0	3,1,0	3,3,0
	HIGH	1,1,1	1,3,1	3,1,1	3,3,1

Table 1. Crossed Design for the experiment

To increase the power of the test, α has been set to 0.1 instead of 0.05 level which is more common [16].

3.5 Experimental results

Due to the type of experiment used, F statistic was applied to obtain the results.
SPSS v. 7.5 software was used for the calculations. Table 2 shows the results for the F-statistic.

Source of variation	Sum of Squares	DF	Mean Square	F	Sig of F
Source of Variation	1.392,132	3	464,044	382,878	0
NT	796,737	1	796,737	657,380	0
NN	553,289	1	553,289	456,514	0
G	42,105	1	42,105	34,741	0
2 Way Interactions	43,184	3	14,395	11,877	0
NT NN	42,105	1	42,105	34,741	0
NT G	0,658	1	0,658	0,543	0,462
NN G	0,421	1	0,421	0,347	0,557
3 Way Interactions	2,132	1	2,132	1,759	0,187
NT NN G	2,132	1	2,132	1,759	0,187
Explained	1.434,447	7	205,350	169,432	0
Redisual	174,526	144	1,212		
Total	1.611,974	151	10,675		

Table 2. Results of the experiment

Comparing these values with $F_{1, 151} = 2.71$, we can ensure that:

Alternative Hypothesis 1: The value of the NT measures affects understandability of SELECT statement. As 657.380>2.71, NT affects results of experiment, so that the alternative hypothesis 1 is valid.
Alternative Hypothesis 2: The value of the NN measures affects understandability of SELECT statement. As 456.514>2.71, NN affects results of experiment, so that the alternative hypothesis 2 is valid.
Alternative Hypothesis 3: The value of the G measures affects understandability of SELECT statement. As 34.741>2.71, G affect to results of experiment, so that the alternative hypothesis 3 is valid.
Alternative Hypothesis 4: Combination of NT and NN measures affects understandability of SELECT statement. As 34.741>2.71, the interaction of NT and NN affects results of experiment, so that, the alternative hypothesis 4 is valid.
Alternative Hypothesis 5: Combination of NT and G affects understandability of SELECT statement. As 0.543<2.71, there is no significant effect of interaction between NT and G
Alternative Hypothesis 6: Combination of NN and G affects understandability of SELECT statement. As 0.421<2.71, there is no significant effect of interaction between NN and G
Alternative Hypothesis 7: Combination of NT, NN and G affects understandability of SELECT statement. As 1.759<2.71, there is no significant effect of interaction between NT, NN and G.
We can conclude that the three kinds of measures proposed have proved to be solid indicators of SQL programs understandability. These three measures are very easy to calculate and could be very useful to predict SQL maintainability.

4 Case study

4.1 General characteristics of the system.

The system is composed of 143 programs developed during a period of one year at the Data Processing Center of the state owned organization (Diputacion of Ciudad Real). The system is a transaction processing system for data maintenance. The programs with embedded SQL are all of small to medium size: each SELECT statement included an average of three tables, two-level nesting and one grouping.
More importantly, the system developed was functionally sound, providing an actual working solution to an actual organizational problem.
One of the most positive aspects of the system is the fact that it was constructed completely by the same team, employing the same methodology and the same developing environment (CA-OpenIngres/4GL).These common factors are advantageous in they can be considered as constants in the analysis, a condition not often encountered in software size research. When they vary, factors such as these can have an obvious impact on system size. Given that these potential contributors may be treated as constant, the degree of confidence adopted in regard to any size relationships supported by the data will consequently be greater [17].

4.2 Data collection

The maintenance process includes adding functionality to the software (adaptive maintenance) and correcting defects discovered in the systems (corrective maintenance).
Our study examined data from a maintenance period beginning with the original installation of the product and ending the product's second release. The required changes varied in magnitude from a simple command line option change to a more complex one. During the maintenance period the programmer recorded the daily effort of product maintenance, descriptions of the faults encountered as a consequence of enhancement activities and the time spent correcting these faults. All these dates are recorded in a specific tool developed in CA-OpenIngres. (see fig. 1 and 2).

Excma. Diputación Provincial de Ciudad Real	Menu Principal
C E N P R I 4/2/2000	Mto. Software Mto. Tablas Generales Gestión del Sistema
Mantenimiento pts 17 Mto. de Software y Soluciones B.D.: Mantenimiento Vers. 1.0	Fechas de CONSULTAS: Inicio: 4/2/2000 Final: 4/2/2000
CbFecha(F6) FechasConsul(F8)	Ejecutar(F9) Fin(F4)

Fig. 2. Screen main of the application

MANTENIMIENTO DE SOFTWARE

C E N P R I
CIUDAD REAL F.CONSULTA:1/1/2000 MANTENIMIENTO
4/1/2000

PENDIENTES: 2 AVISADO: SI

Fecha	Hora	Aplicación	Tipo	Objeto	Asunto	E
10/1	08:02	Nominas	Calculo Subido	Mto. Adaptativo	Adaptación	A
11/1	09:05	Vehículos	Campo Nuevo	Mto. Correctivo	Nuevo Servicio	A
FINALIZADOS: 2						
17/01	08/49	IBI	Ampliar B.D.	Mto. Preceptivo	Arreglar año	F
17/02	10/55	Contabilidad	Cambiar listado	Mto. Correctivo	Ampliar B.D.	F

Consulta(F6) Altas(F7) Modificaciones(F8) Listados(F9)

Fig. 3. Tool window showing an initial task

4.3 Data analysis and results

4.3.1 Descriptive statistics

The general descriptive statistics for each one of the variables are shown in table 3.

Variable	Mean	Variance	S.E. Skew	Min	Max	Std Dev	Skewness
NT	6,042	30,364	0,203	0	22	5,510	1,361
NN	1,923	3,438	0,203	0	6	1,854	0,712
G	0,399	0,241	0,203	0	1	0,492	0,419
TIME	74,364	5.838,865	0,203	1	290	76,413	1,205

Valid observations - 143 Missing observations – 0

Table 3: Descriptive statistics for each measure

4.3.2 Correlation analysis

For the test of correlation we use Pearson´s coefficient statistics and Spearman´s non-parametric correlation to identify potentially relationships between the variable time of maintenance (expressed in minutes) and the measures defined, as well as the relationships that could exist between the same variables. The results are shown in tables 4 and 5.

The statistics of both correlation sets evidence strong significant relationships between the variable time of maintenance and the measure defined, except for measure NG. We can observe that the relationships between the specification of the measured NT, NN and the variable TIME of maintenance are significant.

		NT	NN	G	TIME
Pearson	NT	1,000	0,796	0,280	0,986
	NN	0,796	1,000	0,258	0,881
	G	0,280	0,258	1,000	0,284
	IME	0,986	0,881	0,284	1,000
Sig	NT		0,000	0,001	0,000
	NN	0		0,002	0,000
	G	0,001	0,002		0,001
	TIME	0	0,000	0.001	
N	NT	143	143	143	143
	NN	143	143	143	143
	G	143	143	143	143
	TIME	143	143	143	143

Table 4. Pearson's Correlation Coefficients

		NT	NN	G	TIME
Spearman	NT	1,000	0,799	0,286	0,964
	NN	0,799	1,000	0,243	0,908
	G	0,286	0,243	1,000	0,283
	TIME	0,964	0,908	0,283	1,000
Sig	NT		0,000	0,001	0,000
	NN	0,000		0,003	0,000
	G	0,001	0,003		0,001
	TIME	0,000	0,000	0.001	
N	NT	143	143	143	143
	NN	143	143	143	143
	G	143	143	143	143
	TIME	143	143	143	143

Table 5. Spearman's correlation coefficients.

NT and G are also highly correlated, which is logical because each nesting introduces a table, which usually is different from the table of the previous nesting level. We are conscious that maintenance time can depend on several other different factors than the SELECT characteristics. Some programs have, besides SELECT, other statements like INSERT, DELETE or UPDATE; and also different procedural and visual statements. However, due to the type of the programs involved, we think that these results can be a valid first attempt to characterize SQL programs.
The level of grouping (G) is not correlated with the time of maintenance. We cannot find an answer for this now. More case studies and experiments must be considered in order to explain the influence of grouping in SQL understandability, modifiability and testability.

5 Conclusions and future works

More research is needed into the aspects of software measurement [18], both from theoretical and from practical points of view [19]. We think it is very interesting to dispose of measures for relational databases. We have proposed and validated three types of measures for assessing SQL program maintainability: NT,NN and G. We are also developing some similar measures for other clauses of the SELECT statement such as "HAVING", "WHERE" or "FROM".
More experiments and case studies are needed to confirm these measures as valid indicators for SQL program maintainability. Verification of these metrics in some formal frameworks as [16] or [20] is being carried out.
These measures are not enough to evaluate the maintainability of programs developed with 4GL, so different measures must be proposed for other different "sublanguages" besides data manipulation one. Traditional metrics must be adapted or new metrics must be defined to assess procedural, visual, control, definition and transaction statements.

Acknowledgements

This research has been supported by the MANTICA project CICYT and the European union under grant 1FD97-0168 . We thank paper to Fernando Brito e Abreu and Miguel Goulão for their thoughtful comment which contributed to improve considerably this paper.

References

1. Eisenberg, A. and Melton, J.: SQL: 1999, formerly know as SQL3. SIGMOD Record, 28 ,(1) (1999) 131-138.
2. Card, D.N. and Glass, R.L. : Measuring Software Design Quality. Englewood Cliffs. USA (1990).
3. Pigoski, T.M.: Practical Software Maintenance. Wiley Computer Publishing. New York, USA (1997).
4. Briand, L.C., Morasca, S. and Basili, V.: Property-based software engineering measurement. IEEE Transactions on Software Engineering, 22(1) (1996) 68-85.
5. Fenton, N. and Pfleeger, S. L.: Software Measures: A Rigorous Approach 2nd edition.. Chapman & Hall.London (1997).
6. Melton, A. (ed.) .: Software Measurement. International Thomson Computer Press. London (1996).
7. Sneed, H.M. and Foshag, O. : Measuring Legacy Database Structures. Proc of The European Software Measurement Conference FESMA 98, Antwerp, May 6-8, Coombes, Van Huysduynen and Peeters (eds.), (1998) 199-211.
8. Henderson-Sellers, B. : Object-Oriented Measures - Measures of complexity. Prentice-Hall, Upper Saddle River, New Jersey (1996).
9. Bourque, P. and Côté, V. : An experiment in software sizing with structured analysis measures. Journal of Systems and Software 15 (1991) 159-172.
10. Dolado, J.J.: A Study of the Relationships among Albrecht and Mark II Function Points, Lines of Code 4GL and Effort. J. Systems Software, 37 (1990) 161-173.
11. Verner J. and Tate G.: Estimating Size and Effort in Fourth-Generation Development. IEEE Trans. on Software Engineering (1988) 15-22.
12. ISO/IEC 9126: Information Technology-Software Product Evaluation-Quality characteristics and guidelines for their use, International Organization for standardisation (1994).
13. Li, H.F. and Cheng, W.K. An empirical study of software measures. IEEE Trans. on Software Engineering, 13 (6) (1987) 679-708.
14. Cant, S. N., Jeffery, D.R., and Henderson-Sellers, B.: A concept model of cognitive complexity of element of the programming process. Information and Software Technology. 37(7) (1995) 351-362.
15. Pfleeger, S. L.: Experimental Design and Analysis in Software Engineering. Annals of Software Engineering .JC. Baltzer AG, Science Publishers 1 (1995) 219-253.
16 Briand, L., Bunse, D., Daly, J. and Differding, C.: An experimental comparison of the Maintainability of Object-Oriented and Structured Design Documents, Proc. Int. Conference on Software Maintenance, Harold, M.J. Visaggio, G. (eds), Bari, 1-3 (1997) 130-138.
17. MacDonell, G., Shepperd, J. and Sallis, J.: Measures for Database Systems: An Empirical Study. IEEE Software (1997) 99-107
18. Neil, M.: Measurement as an Alternative to Bureaucracy for the Achievement of Software Quality. Software Quality Journal 3 (2), (1994) 65-78.

19. Glass, R.: The Relationship Between Theory and Practice in Sotfware Engineering. IEEE Software, November, 39(11) (1996) 11-13.
20. Zuse, H.: A framework of software measurement. Walter De Gruyter (1998).

APPENDIX A

TABLE FIELD DESCRIPTION

TABLE DESCRIPTION: employee

```
Name:                   employee
Owner:                  ingres
Created:                24/03/1999 14:08:38
Type:                   user table
Version:                OPING1.2

Column Information:
```

Column Name	Type	Length	Nulls	Defaults	Key Seq
id_emp	varchar	9	no	no	1
namef_emp	varchar	40	no	no	
name_emp	varchar	30	no	no	
direction	varchar	40	no	no	
post_office	varchar	5	no	no	
town	varchar	2	no	no	
village	varchar	3	no	no	
phone	varchar	9	yes	null	
sex	varchar	1	no	no	
status	varchar	1	no	no	
children	varchar	2	no	no	
birth	date		no	no	
birth_town	varchar	2	no	no	
birth_village	varchar	3	no	no	
birth_country	varchar	3	no	no	
number_social_sec	varchar	14	no	no	
number_social_sec1	varchar	6	yes	null	

The table “employee” has 4 rows.

TABLE DESCRIPTION: h_employee

```
Name:                   h_employee
Owner:                  ingres
Created:                20/11/1998 18:05:17
Type:                   user table
Version:                OPING1.2
```

```
Column Information:
                                                                  Key
Column Name                     Type       Length Nulls Defaults Seq
code_service                    varchar         3   no      no
code_subs_service               varchar         2   no      no
id_emp                          varchar         9   no      no
number_fic                      varchar         4   no      no
remainder1                      float           8   no      no
remainder2                      float           8   no      no
remainder3                      float           8   no      no
remainder4                      float           8   no      no
remaindert                      float           8   no      no
remainderf                      float           8   no      no
period                          integer         1   no      no
year                            integer         1   no      no
tipe_hour                       integer         2  yes    null
subtipe                         integer         2  yes    null
key_emp                         varchar        15  yes    null
date_certificate                varchar         6  yes    null
situation                       varchar         1  yes    null
remainderf                      date               yes    null
```

The table “h_employee” has 3 rows.

TABLE DESCRIPTION: control employee

```
Name:                 control_employee
Owner:                ingres
Created:              04/03/1999 13:46:31
Type:                 user table
Version:              OPING1.2

Column Information:
                                                                  Key
Column Name                     Type       Length Nulls Defaults Seq
id_emp                          varchar         9   no      no
number_fic                      varchar         4   no      no
date                            varchar         6   no      no
hour                            varchar         4   no      no
code_incidence                  varchar         2  yes    null
control                         varchar         2  yes    null
status                          varchar         1  yes    null
code_center                     varchar         2  yes    null
tipe                            varchar         2   no      no
```

The table “control_employee” has 72 rows.

SQL STATEMENTS

```
1.-   select hour from control_employee where number_fic='0959' and
      date='181298'

2.-   select number_fic, date, count(hour) as w_number_fic from
      control_employee
      where number_fic='0800' group by number_fic, date
```

```
3.-  select number_fic, date from control_employee where number_fic not
     in
     (select number_fic from control_employee where date='171298' and
     control='SM' and status='A' and hour in (select hour from
     control_employee
     where date='171298' and control='SM' and tipe='A0')) and
     date>'131298'

4.-  select number_fic from control_employee where number_fic not in
     (select Number_fic from control_employee where date='171298' and
     control='SM' and status='A' and hour in (select hour
     from control_employee where date='171298' and control='SM' and
     tipe='A0'))
     and date>'131298' group by number_fic

5.-  select f.name_emp, h.hour, p.key_emp from employee f,
     control_employee h,
     h_employee p where f.id_emp=h.id_emp and h.number_fic=p.number_fic
     and h.date='171298'

6.-  select f.name_emp, p.key_emp from employee f, control_employee h,
     h_employee p where f.id_emp=h.id_emp and h.number_fic=p.number_fic
     and h.date='171298' group by name_emp, key_emp

7.-  select f.name_emp, p.number_fic, p.date, h.date_certificate from
     control_employee, employee f, h_employee h where p.id_emp not in
     (select h.id_emp from control_employee p, employee f, h_employee h
     where p.number_fic=h.number_fic and f.id_emp=h.id_emp and
     p.id_emp=f.id_emp and p.date='171298' and p.control='SM' and
     p.status='A' and f.sex='V' and p.hour in (select hour from
     control_employee p,employee f, h_employee h where
     p.number_fic=h.number_fic and f.id_emp=h.id_emp and
     p.id_emp=f.id_emp                                          and
     p.date='171298' and p.control='SM' and p.tipe='A0' and
     h.remaindert=0)) and p.date='151298' and p.number_fic=h.number_fic
     and p.id_emp=f.id_emp and f.id_emp=h.id_emp

8.-  select f.name_emp, p.number_fic, p.date from control_employee p,
     employee f, h_employee h where p.id_emp not in (select h.id_emp
     from control_employee p, employee f, h_employee h where
     p.number_fic=h.number_fic and f.id_emp=h.id_emp and
     p.id_emp=f.id_emp and p.date='171298' and p.control='SM' and
     p.status='A' and f.sex='V' and p.hour in (select hour from
     control_employee p, employee f, h_employee h where
     p.number_fic=h.number_fic and f.id_emp=h.id_emp and
     p.id_emp=f.id_emp and p.date='171298' and p.control='SM' and
     p.tipe='A0' and h.remaindert=0))and p.date='151298' and
     p.number_fic=h.number_fic and p.id_emp=f.id_emp and
     f.id_emp=h.id_emp
     group by f.name_emp, p.number_fic, p.date
```

Database Migration in WAN Environments: How Can It Earn Good Performance? *

Takahiro Hara Masahiko Tsukamoto Shojiro Nishio

Dept. of Information Systems Eng., Graduate School of Engineering, Osaka University
2-1 Yamadaoka, Suita, Osaka 565-0871, Japan
Email: {hara,tuka,nishio}@ise.eng.osaka-u.ac.jp

Abstract. In broadband networks, appropriate use of database migration can drastically shorten the average transaction processing time of a distributed database system. In previous work, we have proposed a transaction processing method based on database migration, in which we assumed that the network bandwidth is very broad and almost uniform over the whole network. However, WANs consist of various types of networks, including private LANs, narrowband public networks and broadband backbones; thus, that method cannot be applied directly in WAN environments. In this paper, we propose a database migration scheduling method which gives the shortest communication time for database operations, and is suitable for use in WANs. We also prove that the proposed method can find the database migration schedule which gives the shortest communication time for a given access sequence.

1 Introduction

The recent evolution of broadband networks has had a significant impact on the design of database management systems[2, 3, 9, 10]. For distributed database systems in narrowband networks, the minimization of the transmitted data volume is considered as the primary factor in performance improvement. However, in the case of systems in broadband networks, the effective use of the network is a much more significant factor.

Here, the problem is how broadband networks can be used effectively. A possible answer is the migration of databases from site to site through networks. We call such migration *DB-migration*[6]. DB-migration can be performed in a short time period in broadband networks. Therefore, dynamic database relocation using DB-migration can be used for several purposes, including *transaction processing*[6, 7].

In a conventional distributed database system, each database is fixed at a particular site, and a typical database operation is performed through several *operation request messages*. After the message exchange, the operation is validated

* This research was supported in part by Research for the Future Program of Japan Society for the Promotion of Science under the Project "Advanced Multimedia Content Processing (JSPS-RFTF97P00501)".

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 79–91, 2000.

for consistency using the *two-phase commit protocol* (2PC). In this conventional method, transaction processing often requires many message transmissions. On the other hand, if DB-migration is used, a transaction initiation site does not need to exchange messages after it has gathered the necessary databases.

Since the upper limit of data transmission speed is the speed of light, it will be difficult to drastically reduce propagation delay; hence, DB-migration will be more effective as the scale of networks becomes larger. Based on this observation, in our previous work[7], we proposed a transaction processing method which makes effective use of DB-migration. This method chooses the more efficient method of either the conventional database-fixed method or the method using DB-migration. In [7], we also proposed a concurrency control method which considers DB-migration. This method controls the database operations (including DB-migration) which are issued concurrently and prevents transaction processing throughput from deteriorating in environments where data contention is a significant factor.

In our previous work, we assumed a somewhat closed system, such as a system based on ATM (Asynchronous Transfer Mode) virtual LANs in which the network bandwidth in the whole system is almost uniform and very broad. On the contrary, when DB-migration is used in a WAN, a new method is necessary for the following two reasons:

- A WAN consists of various types of networks such as private LANs, narrowband public networks, and broadband backbones. Thus, it is difficult to execute DB-migration between two arbitrary sites in the system. In particular, it is impractical to send a database via narrowband public networks.
- Since DB-migrations can be executed in parallel, it is more desirable to control them in the granularity of the transaction. However, in WAN environments, the system contains a large number of sites, and they each issue various transactions individually. One DB-migration affects the response times of many transactions, and thus, it is very difficult to determine effective DB-migration plans at the transaction level.

In this paper, based on the above facts, we propose a scheduling method for DB-migration which gives the shortest communication time for database operations, a method which is particularly suitable for use in WANs.

Several data relocation methods have been proposed for reducing the average processing time for database operations, such as in [1, 11]. In these methods, if successive read operations to a particular data item are issued at site S_j while no write operations are issued by other sites to the item, its copy is created at S_j. Obviously, this results in a large number of copies of small data items in the system. These methods are impractical because the location management overhead of replicas is very large and because these methods require many message transmissions to maintain consistency among the replicas and to check database constraints. On the other hand, since our proposed method dynamically relocates the databases (large data sets) using DB-migration without creating their copies, the problems associated with conventional methods can be resolved.

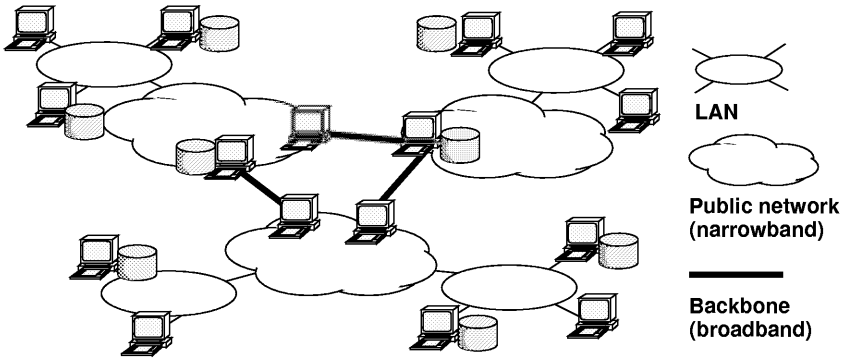


Fig. 1. A distributed database system in a WAN.

The remainder of the paper is organized as follows. In section 2, we propose a scheduling method for DB-migration which gives the minimum communication time for database operations. In section 3, we prove the correctness of our proposed method. Finally, in section 4, we summarize the paper.

2 The Optimal Scheduling of DB-migration

In this section, first we describe the assumed system environment. Then, we propose a DB-migration scheduling method which is suitable for WAN environments and gives the shortest communication time for database operations.

2.1 The system model

The system environment is assumed to be a distributed database system which is constructed in a WAN environment such as the one shown in Figure 1. The WAN consists of three types of networks: broadband LANs, narrowband public networks, and broadband backbones. All communications from sites in LANs to sites in backbones are performed via at least one public network. In Figure 1, there exist two backbone sub-networks that consist of two and three sites, respectively. Although the lines representing backbones are not so long in Figure 1, the backbones they represent are usually long, and sometimes very long, e.g., from San Francisco to Tokyo.

In this network, we also make the following assumptions:

- A unique *site identifier*, S_i $(i = 1, 2, \cdots)$, is assigned to each site in the system.
- DB-migrations are performed only in backbones. This is assumed for the following reasons: (i) DB-migration in LANs has little benefit for the system performance because propagation delays for database operations are

very small in LANs, and (ii) DB-migration in public networks is impractical because it takes so much time to perform.

- $D_{i,j}$ denotes the communication time necessary for performing DB-migration between sites S_i and S_j in backbone sub-networks. This is determined based on the bandwidth and the propagation delay between the two sites. $d_{i,j}$ denotes the communication time necessary for performing a message transmission of small data volume between sites S_i and S_j. This is equal to the propagation delay between the two sites. We also assume that the following relations are satisfied among three arbitrary sites, S_i, S_j, and S_k:

$$\begin{aligned} D_{i,k} &\leq D_{i,j} + D_{j,k} \\ d_{i,k} &\leq d_{i,j} + d_{j,k}. \end{aligned} \tag{1}$$

  These assumptions are realistic considering usual network systems.
- Very high-speed database access is achievable using main memory database system techniques [4, 5]. The migration of indexes can also be carried out by applying the mechanism we proposed in [8].
- A database replica is not created. This is accomplished by deleting the database at the database sender site after the DB-migration is completed. Here, while extra overhead is inevitable in order to maintain consistency between the primary database and each replica, database replication reduces the number of message transmissions for database operations. Several features of the system determine whether or not it is effective to make a replica. Database replication in WANs with DB-migration is part of our future work.
- An *access sequence* of a database is expressed as a list of site identifiers, $\mathtt{A} = [S_{i1}, \cdots, S_{im}]$, so that the list represents the temporal order of sites which issue access (operation) requests to the database. Moreover, a subsequence of an access sequence is called an *access subsequence*, a point (an element) which is the first or the last in the access subsequence is called a boundary point, and a point which is in the access subsequence and is not a boundary point is called an *intermediate point*.
- A list of site identifiers is called a *site sequence*. A schedule of DB-migration for an access sequence of m database operations is expressed as a list of m site sequences, $\mathtt{S} = [\mathbf{S}_1, \cdots, \mathbf{S}_m]$. Here, $\mathbf{S}_j$ $(1 \leq j \leq m)$ represents that the database successively migrates to sites in the order represented by $\mathbf{S}_j$ before the j-th operation in the relevant access sequence is executed. Moreover, a partial list of a schedule is called a *partial schedule*.
- A cost, $T(S_I, \mathtt{A}, \mathtt{S})$, is the total communication time required when access sequence $\mathtt{A}$ for a database at site S_I is executed based on schedule $\mathtt{S} = [\mathbf{S}_1, \cdots, \mathbf{S}_m]$. For the purpose of simplicity, we call it the cost of schedule $\mathtt{S}$. Let k_j denote the number of list elements in $\mathbf{S}_j$, where $\mathbf{S}_j = [S_{j1} \cdots, S_{jk_j}]$ $(1 \leq j \leq m)$. The cost of schedule $\mathtt{S}$ is expressed by the following equation:

$$T(S_I, \mathtt{A}, \mathtt{S}) = \sum_{j=1}^{m} \left\{ \sum_{l=0}^{k_j - 1} D_{jl,j(l+1)} + d_{ij,jk_j} \right\}. \tag{2}$$

  Here, $S_{(1)(0)}$ denotes S_I, and S_{j0} $(2 \leq j \leq m)$ denotes $S_{(j-1)k_{(j-1)}}$.

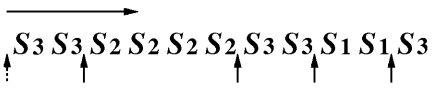


Fig. 2. Access sequence and turning points.

In this paper, for a given access sequence, we show how to determine a schedule of DB-migration so that it gives the shortest communication time for database operations. Here, a schedule should be separately determined for each database because the characteristics of WANs shown in section 1 make it difficult to control DB-migrations in the granularity of the transactions. As mentioned before, DB-migration is only performed among sites which are in the same backbone sub-network. Thus, we address the problem of minimizing the communication time necessary for database operations which are issued from sites in the backbone sub-network. Strictly speaking, database operations are issued from every site in the WAN, and they are executed at sites in the backbone sub-network.

2.2 Scheduling of DB-migration

In this subsection, we propose a DB-migration scheduling method which minimizes the communication time for database operations.

Figure 2 show an example of an access sequence. In the figure, each upward arrow indicates a point where the access initiation site changes in the access sequence, and we call this point a *turning point*. The site just after a turning point is called a *turning point site*. If the first access in the access sequence is issued from a site at which the database does not initially reside, a point before the first access also becomes a turning point (See the dotted arrow in Figure 2).

Composing the optimal schedule by checking every possible DB-migration at every access point is an extremely complex problem that cannot be practically solved. In this paper, we propose a scheduling algorithm, called the *Cost-Optimal DB-migration scheduling (CODB)* algorithm, which gives the shortest communication time with less computation.

The CODB algorithm:

1. A root node representing a site at which the database initially resides is created, and its *node value* is set to 0. This node is called a *level 1 node*.
2. The given access sequence is scanned. If a turning point is found, for each node of the current maximum level (level i), at most N child nodes representing sites in the backbone sub-network are created such that the sites represented by the child nodes give shorter or equal propagation delay to the turning point site than the parent node. Here, N denotes the total number of sites which exist in the backbone sub-network. For example, if a parent

node represents the turning point site, the only node created is a child node representing the turning point site.
The newly created nodes are called *level* $i+1$ *nodes*.

3. The node value of a child node representing S_j is set to the value formed by adding $D_{p,j} + L \cdot d_{t,j}$ to the node value of the parent. Furthermore the site that the parent node represents is denoted by S_p, and the turning point site is denoted by S_t. L denotes the number of elements between the turning point and next turning point in the given access sequence. If it is the last turning point in the access sequence, L denotes the number of elements between it and the last point.
If there are no more turning points in the access sequence, go to step 5. Otherwise, go to step 4.
4. For each set of level $i+1$ nodes representing each of N sites, only one node which gives the minimum node value is selected; the other nodes are deleted. If there are several nodes which give the minimum node value, only one node among them is chosen, and the others are deleted.
After this process is completed, return to step 2.
5. Among all level $i+1$ nodes, a node which gives the minimum node value is selected. Then, scanning a path from the root to the selected node, a schedule of DB-migration is determined so that the database resides at the site corresponding to the most recent turning point. (End of the algorithm.)

Let M denote the number of turning points in the access sequence. At the end of the algorithm, a tree has been constructed such that its root represents the site at which the database initially resides, the total number of nodes at each level except for the maximum level is at most N, and its depth is M. The increment $(D_{p,j} + L \cdot d_{t,j})$ between the node value of a parent and that of each child indicates the communication time necessary for making the database migrate to site S_j and executing database operations between the turning point and the next turning point in the access sequence. In other words, the node value indicates the communication time necessary for making the database migrate from the initial site to the site which the node represents via sites represented by intermediate nodes and for executing database operations between the first point and the turning point which follows the current turning point in the access sequence.

In step 4, at each level except for the maximum level, due to the algorithm deletes all the nodes that do not give the minimum node value among nodes representing the same site, N nodes are left after comparing node values $\sum_{i=1}^{N}(i-1)$ times. At the maximum level, there are $\sum_{i=1}^{N} i$ nodes and $\sum_{i=1}^{N} i-1$ comparisons of node values being performed. Therefore, the computational complexity of the algorithm is $O(N^2M)$.

2.3 Example of the algorithm execution

In this subsection, we show an example of executing our proposed CODB algorithm. First, for the system environment, a backbone sub-network as shown

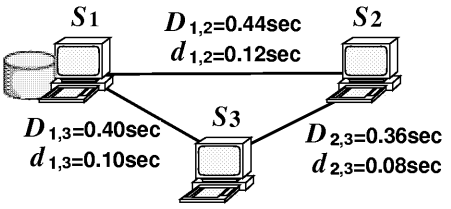


Fig. 3. An example of a system.

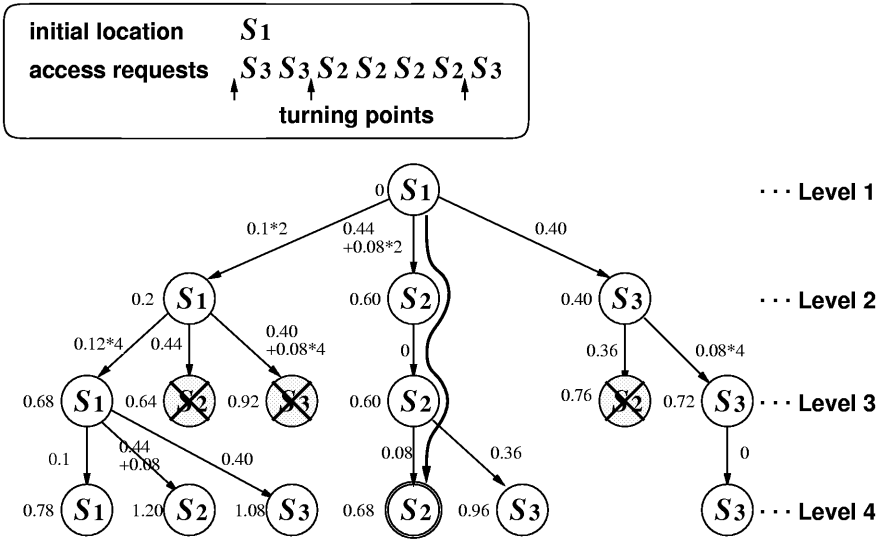


Fig. 4. An example of executing the CODB algorithm.

in Figure 3 is assumed. Here, $D_{1,2}$, $D_{2,3}$, and $D_{1,3}$ are 0.44 seconds, 0.36 seconds, and 0.40 seconds, respectively, and $d_{1,2}$, $d_{2,3}$, and $d_{1,3}$ are 0.12 seconds, 0.08 seconds, and 0.10 seconds, respectively.

Given that the initial location of the database is S_1 and the access sequence is $\mathtt{A} = [S_3, S_3, S_2, S_2, S_2, S_2, S_3]$ (See the upper part of Figure 4), a schedule of DB-migration is determined by the CODB algorithm.

The result of executing the algorithm is shown in the lower part of Figure 4. In the result graph, the node value is shown to the left of each node, and the incremental value between a parent and a child is shown along each edge. At each level, the nodes representing the sites which give shorter propagation delay are created, and the communication times necessary for making the database migrate to the representing sites and executing the database operations are calculated.

At level 3, the nodes which do not give the minimum node value among nodes representing the same site are deleted. Finally, at level 4, the site which gives the minimum node value is determined. As a result, since the path (S_1, S_2, S_2, S_2) gives the minimum total node value, DB-migrations are performed so that the database resides at the corresponding site at each turning point. Therefore, the schedule makes the database migrate to S_2 at the first turning point and does not perform DB-migration after that ($\mathtt{S} = [[S_2], [S_2], [S_2], [S_2], [S_2], [S_2], [S_2]]$).

3 Correctness of the CODB Algorithm

In this section, we prove that the CODB algorithm finds the DB-migration schedule which gives the shortest communication time for a given access sequence.

First, we show a characteristic of turning points.

Theorem 1:

There exists an optimal schedule where each DB-migration occurs at a turning point.

Proof of Theorem 1:

First, we show the following lemma.

Lemma 1:

When DB-migrations are performed at a point in an access subsequence and performing the migrations at an intermediate point gives the shortest communication time, performing the migrations at the boundary point also gives the shortest communication time.

Proof of Lemma 1:

Assume an access subsequence $\mathtt{A}_1$ consisting of b database operations from site S_i. If m DB-migrations from S_{j0} (the initial location) to S_{jm} are performed after the a-th operation as shown in Figure 5, the schedule $\mathtt{S}_1$ is expressed as $\mathtt{S}_1 = [[S_{j0}], \cdots, [S_{j0}], [S_{j1}, \cdots, S_{jm}], [S_{jm}], \cdots, [S_{jm}]]$. The first through a-th elements are all S_{j0}, the $a+1$-th element is a list consisting of m elements from S_{j1} to S_{jm}, and the rest of the elements are all S_{jm}. Here, S_{jl} $(0 < l \leq)$ denotes the database receiver site of the l-th DB-migration. Based on Equation (2), the cost of schedule $\mathtt{S}_1$, $T(S_{j0}, \mathtt{A}_1, \mathtt{S}_1)$, is expressed by the following equation:

$$T(S_{j0}, \mathtt{A}_1, \mathtt{S}_1) = \sum_{k=0}^{m-1} D_{jk,j(k+1)} + a \cdot d_{i,j0} + (b-a) d_{i,jm}. \tag{3}$$

On the right-hand side of this equation, the second and the third terms include the variable a, and the minimum cost is given when a $(0 \leq a \leq b)$ is the following:

$$\begin{array}{ll} 0 & (\text{where} \quad d_{i,j0} > d_{i,jm}) \\ b & (\text{where} \quad d_{i,j0} < d_{i,jm}) \\ \text{any value} & (\text{where} \quad d_{i,j0} = d_{i,jm}) \end{array}$$

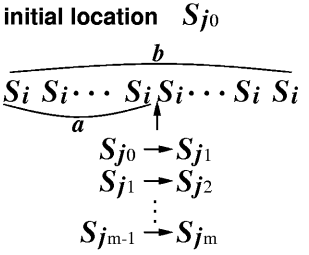


Fig. 5. Notations for the proof of Lemma 1.

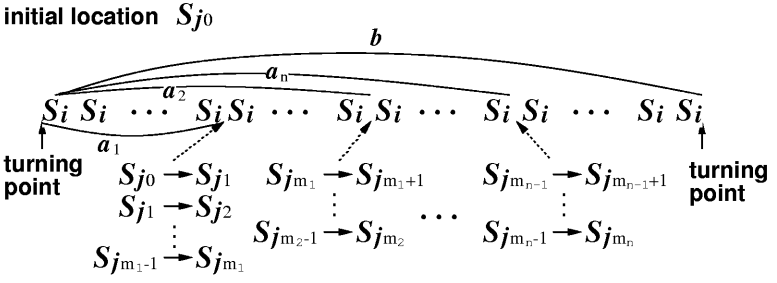


Fig. 6. Notations for the proof of Theorem 1.

If the cost, i.e., the communication time, is minimized by performing DB-migrations at an intermediate point, the condition $d_{i,j0} = d_{i,jm}$ is satisfied, and in this case, DB-migrations at every point including the boundary point also give the same cost. Thus, when the DB-migrations at an intermediate point give the shortest communication time, the migrations at the preceding or following boundary point also give the shortest communication time. □

From Lemma 1, in an access subsequence from 0 to b, either of the boundary points is the optimal a value independently of $d_{i,j0}$ and $d_{i,jm}$.

Next, as shown in Figure 6, an access subsequence consisting of b database operations from site S_i is assumed. In the sequence, both of the boundary points are turning points and no other turning point exists. DB-migrations are performed at n points in the sequence, and S_{j0} denotes the initial location of the database. a_k elements exist between the first point and k-th migration point, and $m_k - m_{k-1}$ successive DB-migrations from $S_{jm_{k-1}}$ to S_{jm_k} are performed at the k-th point ($m_0 = 0$). Here, S_{jl} $(0 < l \leq m_n)$ denotes the database receiver site of the l-th DB-migration in the access subsequence.

First, in the area between the first point and a_2-th point, we examine the value of a_1 which gives the shortest communication time necessary for the first

a_2 database operations. From Lemma 1, the optimal value of a_1 is either 0 or a_2. If the optimal value of a_1 is 0, DB-migrations from S_{j0} to S_{jm_1} are performed at the first point, i.e., the turning point, in the access subsequence. If the optimal value of a_1 is a_2, DB-migrations from S_{j0} to S_{jm_1} are performed at a_2, and as a result, m_2 DB-migrations from S_{j0} to S_{jm_2} are performed at a_2.

Next, we examine the optimal value of a_2. If the optimal value of a_1 is 0, then $m_2 - m_1$ DB-migrations from S_{jm_1} to S_{jm_2} are performed at a_2 $(0 \leq a_2 \leq a_3)$, and the optimal a_2 value gives the shortest communication time necessary for the first a_3 database operations. If the optimal value of a_1 is a_2, then m_2 DB-migrations from S_{j0} to S_{jm_2} are performed at a_2 $(0 \leq a_2 \leq a_3)$, and the optimal a_2 value still gives the shortest communication time necessary for a_3 database operations. In both cases, from Lemma 1, either $a_2 = 0$ or $a_2 = a_3$ gives the shortest communication time.

It is inductively shown that the communication time for a_{k+1} $(2 < k < n)$ operations becomes shortest when each of the first m_k DB-migrations is performed either at $a_k = 0$ or $a_k = a_{k+1}$. Finally, the communication time for b database operations becomes shortest when each of m_n DB-migrations is performed either at the first point or the last point in the access subsequence. In any case, all DB-migrations are performed at the turning point(s).

In conclusion, for an access subsequence in which both of the boundary points are turning points and no other turning point exists, the optimal schedule can be determined by just considering the boundary turning points. Therefore, for a given access sequence, there exists an optimal schedule which does not include DB-migrations at any point excepting turning points. □ (End of proof.)

Theorem 1 guarantees that only DB-migrations at turning points are considered in step 2 of the CODB algorithm.

Theorem 2:

There exists an optimal schedule which does not include more than one DB-migration at each turning point.

Proof of Theorem 2:

When a database migrates from the initial site S_{j0} to site S_{j2} at a turning point, let us suppose the following two cases: (i) the database migrates first to site S_{j1} and then to S_{j2}, and (ii) the database migrates directly to S_{j2}. From the assumption in section 2.1, the following relation is satisfied between the two cases:

$$D_{j0,j1} + D_{j1,j2} \geq D_{j0,j2}. \tag{4}$$

It is inductively shown that the direct DB-migration from S_{j0} to S_{j2} gives the shorter communication time than the case in which the database migrates from S_{j0} to S_{j2} via more than one DB-migration.

Therefore, the optimal schedule can be determined by considering only the cases in which the database migrates to possible sites by a direct DB-migration at each turning point. □ (End of proof.)

Theorem 2 guarantees that the communication times are calculated only when the database migrates to possible sites by a direct DB-migration.

In the following, let $\mathtt{a} + \mathtt{b}$ denote the append operation of lists $\mathtt{a}$ and $\mathtt{b}$.

Theorem 3:

The optimal schedule does not include any DB-migrations to a site which gives a propagation delay to the turning point site that is longer than the propagation delay between the turning point site and the site at which the database currently resides.

Proof of Theorem 3:

At the beginning of the proof, we show the following lemma.

Lemma 2:

When access sequence $\mathtt{A}$ *of the database which initially resides at* S_I *consists of access subsequences* a_1 *and* a_2 ($\mathtt{A} = a_1 + a_2$)*, and when schedule* $\mathtt{S}$ *for* $\mathtt{A}$ *consists of partial schedules* $\mathtt{s}_1$ *and* $\mathtt{s}_2$ ($\mathtt{S} = \mathtt{s}_1 + \mathtt{s}_2$) *which are for* $\mathtt{a}_1$ *and* $\mathtt{a}_2$*, respectively, the following equations hold:*

$$T(S_I, \mathtt{A}, \mathtt{S}) = T(S_I, \mathtt{a}_1 + \mathtt{a}_2, \mathtt{s}_1 + \mathtt{s}_2) = T(S_I, \mathtt{a}_1, \mathtt{s}_1) + T(S_J, \mathtt{a}_2, \mathtt{s}_2).$$

Here S_J denotes the location of the database after partial schedule $\mathtt{s}_1$ is executed.

Proof of Lemma 2:

When $\mathtt{s}_1$ and $\mathtt{s}_2$ consist of m_1 and $m - m_1$ site sequences respectively, the equations below follow from Equation (2):

$$\begin{aligned} T(S_I, \mathtt{A}, \mathtt{S}) &= \sum_{j=1}^{m} \left\{ \sum_{l=0}^{k_j - 1} D_{jl,j(l+1)} + d_{ij,jk_j} \right\} \\ &= \sum_{j=1}^{m_1} \left\{ \sum_{l=0}^{k_j - 1} D_{jl,j(l+1)} + d_{ij,jk_j} \right\} + \sum_{j=m_1+1}^{m} \left\{ \sum_{l=0}^{k_j - 1} D_{jl,j(l+1)} + d_{ij,jk_j} \right\} \\ &= T(S_I, \mathtt{a}_1, \mathtt{s}_1) + T(S_J, \mathtt{a}_2, \mathtt{s}_2). \end{aligned}$$

□

Now, let us consider schedule $\mathtt{S} = \mathtt{s}_1 + [S_{j1}] + \mathtt{s}_2$ for access sequence $\mathtt{A} = \mathtt{a}_1 + [S_i] + \mathtt{a}_2$ with initial database location S_I (S_i is a turning point site). $\mathtt{S}$ includes a DB-migration to site S_{j1} at a turning point just after $\mathtt{a}_1$ so that the migration satisfies $d_{i,j0} < d_{i,j1}$. $\mathtt{s}_1$ and $\mathtt{s}_2$ denote the partial schedules for access subsequences $\mathtt{a}_1$ and $\mathtt{a}_2$, respectively, and S_{j0} denotes the location of the database after executing $\mathtt{s}_1$.

Let us also consider schedule $\mathtt{S'} = \mathtt{s}_1 + [S_{j0}] + \mathtt{s'}_2$ which is equal to $\mathtt{S}$ except that the DB-migration is not performed at the turning point just after $\mathtt{a}_1$ but is instead performed at the next turning point. If $\mathtt{S}$ includes a DB-migration at the turning point immediately after $\mathtt{a}_1$, in $\mathtt{S'}$, that migration is performed immediately following a DB-migration from S_{j0} to S_{j1}. $\mathtt{s'}_2$ denotes a partial schedule after the turning point in $\mathtt{S'}$.

The costs of $\mathtt{S}$ and $\mathtt{S'}$ are expressed by the following equations:

$$\begin{aligned} T(S_I, \mathtt{A}, \mathtt{S}) &= T(S_I, \mathtt{a}_1 + [S_i] + \mathtt{a}_2, \mathtt{s}_1 + [S_{j1}] + \mathtt{s}_2) \\ &= T(S_I, \mathtt{a}_1, \mathtt{s}_1) + D_{j0,j1} + n \cdot d_{i,j1} + T(S_{j1}, \mathtt{a}_2, \mathtt{s}_2) \\ T(S_I, \mathtt{A}, \mathtt{S'}) &= T(S_I, \mathtt{a}_1 + [S_i] + \mathtt{a}_2, \mathtt{s}_1 + [S_{j0}] + \mathtt{s'}_2) \\ &= T(S_I, \mathtt{a}_1, \mathtt{s}_1) + n \cdot d_{i,j0} + D_{j0,j1} + T(S_{j1}, \mathtt{a}_2, \mathtt{s}_2). \end{aligned}$$

Here n denotes the number of database operations between the turning point and its next turning point ($n > 0$). Since $d_{i,j0} > d_{i,j1}$, $T(S_I, \mathtt{A}, \mathtt{S}) > T(S_I, \mathtt{A}, \mathtt{S'})$. □ (End of proof.)

Theorem 3 guarantees that the algorithm creates only child nodes representing sites which give shorter propagation delays to the turning point site than the site which the parent node represents.

Theorem 4:

The optimal schedule does not include partial schedule $\mathtt{s}$ *which finally locates the database at* S_{j0} *when there exists another partial schedule* $\mathtt{s'}$ *which locates the database at* S_{j0} *and costs less than* $\mathtt{s}$.

Proof of Theorem 4:

Let us consider the schedule $\mathtt{S} = \mathtt{s}_1 + \mathtt{s} + \mathtt{s}_2$ for access sequence $\mathtt{A} = \mathtt{a}_1 + \mathtt{a} + \mathtt{a}_2$ with initial database location S_I. Let $\mathtt{s}_1$, $\mathtt{s}$, and $\mathtt{s}_2$ denote partial schedules for access subsequences $\mathtt{a}_1$, $\mathtt{a}$, and $\mathtt{a}_2$, respectively. Let S_{j0} and S_{j1} denote the database locations after executing $\mathtt{s}_1$ and $\mathtt{s}_2$, respectively.

Let us also consider another schedule $\mathtt{S'} = \mathtt{s}_1 + \mathtt{s'} + \mathtt{s}_2$ where $\mathtt{s'}$ locates the database at S_{j1} and $T(S_{j0}, \mathtt{a}, \mathtt{s}) > T(S_{j0}, \mathtt{a}, \mathtt{s'})$ is satisfied. Here, the following equations and inequation hold:

$$\begin{aligned} T(S_I, \mathtt{A}, \mathtt{S}) &= T(S_I, \mathtt{a}_1, \mathtt{s}_1) + T(S_{j0}, \mathtt{a}, \mathtt{s}) + T(S_{j1}, \mathtt{a}_2, \mathtt{s}_2) \\ &> T(S_I, \mathtt{a}_1, \mathtt{s}_1) + T(S_{j0}, \mathtt{a}, \mathtt{s'}) + T(S_{j1}, \mathtt{a}_2, \mathtt{s}_2) = T(S_I, \mathtt{A}, \mathtt{S'}). \end{aligned}$$

□ (End of proof.)

Theorem 4 guarantees that the nodes which do not give the minimum node value among nodes representing the same site are deleted at each level in step 4 of the CODB algorithm.

As a consequence, the following corollary is derived.

Corollary:

The optimal schedule which gives the shortest communication time is determined by the CODB algorithm.

4 Conclusion

We have proposed a new algorithm for scheduling DB-migration which minimizes the communication time necessary for database operations in backbone sub-networks, assuming that it is difficult to control DB-migration in the granularity of the transaction and that it is also difficult to execute DB-migrations through networks except for backbones in a WAN environment. The proposed CODB algorithm can determine the optimal schedule with a computational complexity of $O(N^2M)$. Here, N denotes the number of sites in the backbone sub-network and M denotes the number of turning points in the given access sequence.

The CODB algorithm can be directly applied in closed environments such as ATM virtual LANs. As part of our future work, we are planning to compare the performance of the CODB algorithm and the method proposed in [7] in

such environments. We are also planning to implement the CODB algorithm on a practical platform to evaluate the algorithm in several areas including the protocol overhead. Furthermore, we should consider the scheduling of replica allocation in order to minimize the communication time necessary for database operations in environments where database replicas exist.

References

1. Acharya, S. and Zdonik, S.B., "An Efficient Scheme for Dynamic Data Replication," Technical Report CS-93-43, Brown University, Providence, RI (Sept. 1993).
2. Banerjee, S., Li, V.O.K., and Wang, C., "Distributed Database Systems in High-Speed Wide-Area Networks," *IEEE Journal on Selected Areas in Commun.*, Vol.11, No.4, pp.617-630 (May 1993).
3. Banerjee, S. and Chrysanthis, P.K., "Network latency optimizations in distributed database systems," *Proc. of 14th Int'l Conf. on Data Engineering (ICDE'98)*, pp. 532-540 (Feb. 1998).
4. DeWitt, D., Katz, R., Olken, F., Shapiro, L., Stonebraker, M., and Wood, D., "Implementation Techniques for Main Memory Database Systems," *Proc. of ACM SIGMOD'84*, pp.1-8 (June 1984).
5. Garcia-Molina, H., Lipton, R.J., and Valdes, J., "A Massive Memory Machine," *IEEE Trans. Comput.*, Vol.C-33, pp.391-399 (May 1984).
6. Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S., "Database Migration: A New Architecture for Transaction Processing in Broadband Networks," *IEEE Trans. on Knowledge and Data Engineering*, Vol.10, No.5, pp.839-854 (Sept./Oct. 1998).
7. Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S., "DB-MAN: A Distributed Database System Based on Database Migration in ATM Networks," *Proc. of 14th Int'l Conf. on Data Engineering (ICDE'98)*, pp.522-531 (Feb. 1998).
8. Hara, T., Harumoto, K., Tsukamoto, M., and Nishio, S., "Main Memory Database for Supporting Database Migration," *Proc. of IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing (IEEE PACRIM '97)*, Vol.1, pp.231-234 (Aug. 1997).
9. Herman, G., Gopal, G., Lee, K., and Weinrib, A., "The Datacycle Architecture for Very High Throughput Database Systems," *Proc. of ACM SIGMOD'87*, pp.97-103 (1987).
10. Stonebraker, M., Aoki, P.M., Devine, R., Litwin, W., and Olson, M., "Mariposa: A New Architecture for Distributed Data," *Proc. of 10th Int'l Conf. on Data Engineering (ICDE'94)*, pp.54-65 (1994).
11. Wolfson, O. and Jajodia, S., "Distributed Algorithms for Dynamic Replication of Data," *Proc. of ACM PODS'92*, pp.149-163 (June 1992).

Dynamic Reconfiguration Algorithm: Dynamically Tuning Multiple Buffer Pools*

Patrick Martin[1], Hoi-Ying Li[1], Min Zheng[1], Keri Romanufa[2], and Wendy Powley[1]

[1]Department of Computing and Information Science, Queen's University, Kingston, Ontario Canada K7L 3N6
{martin, li, zheng, wendy}@cs.queensu.ca

[2]IBM Toronto Laboratory, Toronto, Ontario Canada M3C 1H7
keri@ca.ibm.com

Abstract. The tasks of configuring and tuning large database management systems (DBMSs) have always been both complex and time-consuming. They require knowledge of the characteristics of the system, the data, and the workload, and of the interrelationships between them. The buffer pools, because they exist to reduce the number of disk accesses performed by a transaction, are a key resource in a DBMS. Current DBMSs, such as DB2 Universal Database, divide the buffer area into a number of independent buffer pools and database objects (tables and indices) are assigned to a specific buffer pool. The size of each buffer pool is set by configuration parameters and page replacement is local to each buffer pool. Tuning the size of the buffer pools to a workload is crucial to achieving good performance. In this paper we describe a self-tuning algorithm, called the *Dynamic Reconfiguration algorithm* (DRF), for managing the buffer pools in a DBMS and we present the results of a set of experiments to investigate the performance of an implementation of the algorithm for DB2 Universal Database.

1. Introduction

The tasks of configuring and tuning a database management system (DBMS), which are currently primarily manual exercises, are necessary to ensure acceptable performance. They require knowledge of the characteristics of the system, the data, the workload, and of the interrelationships between them. The increasing complexity of the DBMSs and their workloads means that manually managing the performance of a DBMS via direct adjustment of low-level system parameters is becoming impractical.

The buffer area used by a DBMS is particularly important to system performance because effective use of the buffers can reduce the number of disk accesses per-

* This research is supported by IBM Canada Ltd., NSERC (National Science and Engineering Research Council) and CITO (Communications and Information Technology Ontario).

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 92–101, 2000.

formed by a transaction. Current DBMSs, such as DB2 Universal Database (DB2/UDB) [5], divide the buffer area into a number of independent *buffer pools* and database objects (tables and indices) are assigned to a specific buffer pool. The size of each buffer pool is set by configuration parameters and page replacement is local to each buffer pool. Tuning the size of the buffer pools to a workload is therefore crucial to achieving good performance.

In this paper we present a self-tuning algorithm for multiple buffer pools. Our *Dynamic Reconfiguration algorithm* (*DRF*) is based on the concept of *goal-oriented resource management*, which allows administrators to specify their expectations, or goals, for performance while leaving it up to the system to decide how to achieve those goals [4].

DRF has been implemented and tested with DB2/UDB. We show the results of a set of experiments using DRF to tune the buffer pools in DB2/UDB for an OLTP workload as typified by the TPC-C benchmark [9].

The remainder of the paper is structured as follows. Section 2 discusses related work. Section 3 describes our system model and our self-tuning algorithm. Section 4 presents a set of experiments we conducted to evaluate our algorithm. Section 5 summarizes the paper and presents ideas for future work.

2. Related Work

Goal-oriented resource management techniques have been proposed for distributed computing [7] as well as for database management systems [2], [3], [4]. Goal-oriented mechanisms typically use an iterative approach since it is difficult to accurately predict the buffer allocation required to produce a specific response time. Each iteration is a process of observing actual response times, estimating new allocations and then adjusting the allocations. If the mechanism is functioning correctly, then it will converge to an allocation that either achieves the set of performance goals or comes as close as possible to achieving them. To the best of our knowledge, three previous goal-oriented buffer tuning algorithms have appeared in the literature: dynamic tuning [4], fragment fencing [2] and class fencing [3].

An important difference between DRF and the three previous approaches is the assumed model of buffer pool organization. The previous approaches all use a *transaction-oriented model*, that is, they assume that buffer pools are organized based on workload classes. DRF, on the other hand, uses a *data-oriented* model that assumes that buffer pools are organized based on database objects. In our model, the buffer pool pages used by a transaction class are not likely to be in a single buffer pool but instead spread out over several buffer pools.

Another difference between our work and the previous research is how the approaches are validated. The other algorithms are analyzed using simulation studies. In this paper we present an experimental study of an implementation of DRF for DB2/UDB.

3. Dynamic Reconfiguration Algorithm

The aim of our Dynamic Reconfiguration algorithm is to provide an allocation of buffer pages to buffer pools such that the performance goals of the transaction classes using the database are met. The DBA provides an average response time goal for each of the transaction classes in the system workload. We assume that a *transaction class* is a collection of transactions with the same requirements, that is, they access the same set of data objects and have the same performance goals. For example, in the TPC-C benchmark [7], which typifies an OLTP application, we can represent each type of transaction with its own class. So the transaction classes for TPC-C are New Order, Payment, Order Status, Delivery and Stock Level.

DRF compares current performance measurements with the performance goals of each transaction class. If one or more of the transaction classes are not meeting their goals then DRF attempts to find a reallocation of buffer pages to buffer pools such that all the classes meet their goals. The performance of the DBMS relative to the transaction classes' goals is measured by the *Achievement Index* for each transaction class T_i, which is given by

$$AI_i = \frac{Goal\ \ Average\ \ Response\,Time for T_i}{Actual\ Average\ Response\,Time for T_i}$$

If $AI_i < 1$ then class T_i is not achieving its goal. If $AI_i \geq 1$ then class T_i is meeting or exceeding its goal. DRF tries to converge to a situation where each AI_i is close to 1. DRF determines a reallocation of buffer pool pages in favour of the transaction class with the smallest *AI*. We call this class the *target* transaction class for the tuning session.

An iteration of DRF reallocates a fixed number of pages from one buffer pool to another. Adding pages to a buffer pool can increase the hit rate of the buffer pool, which in turn reduces the response time of transactions using that buffer pool since there are, on average, fewer accesses to the disk. The effect of a reallocation is estimated using the cost estimate equations described below.

The *target* buffer pool for a reallocation is the buffer pool that, when given more pages, provides the largest performance improvement to the target class. The *source* buffer pool for a reallocation is the buffer pool that, when relieved of pages, has the smallest negative impact on the performance of the target class. DRF repeats the reallocation exercise until all transaction classes meet their goals or no further improvement in performance can be achieved.

We assume a buffer pool model similar to the buffer pool organization used in DB2/UDB [5]. Buffer memory is partitioned into a number of independent buffer pools and database objects (tables and indices) are assigned to specific buffer pools when the system is configured. An object's pages are moved between disk and it's designated buffer pool. The size of each buffer pool is set by configuration parameters and page replacement is local to each buffer pool.

An access to a buffer pool by a transaction is called a *logical read*. If the page required by the logical read is already in the buffer pool then the DBMS can satisfy

the request immediately. If the required page is not in the buffer pool then it must be retrieved from the disk, which is called a *synchronous read*. The proportion of logical reads that require a disk access is called the buffer pool's *miss rate*.

If there are no free clean pages to hold the new page then a dirty page, that is a page with updates, must be selected for replacement and written back to disk in order to make room for the new page. This write is called a synchronous write. The DBMS may use background tasks to enhance buffer pool performance by performing *asynchronous I/O*, which is system-initiated data transfer between disk and the buffer pools. *I/O servers* are background tasks that prefetch pages into the buffer pools. We say that I/O servers perform *asynchronous reads*. *I/O cleaners* are background tasks that write dirty pages back to disk. We say that I/O cleaners perform *asynchronous writes*

Cost Estimate Equations

A number of the cost estimates described below use the least squares approximation [6] curve fitting technique to calculate values for arbitrary memory sizes. In these cases we collect various performance statistics for three different buffer pool sizes. For each configuration, we collect for each buffer pool: the number of logical reads, the number of physical reads, the number of asynchronous reads, the number of physical writes, the number of asynchronous writes, the average cost of a physical read and the average cost of a physical write.

An application using the DBMS is characterized by a set of transaction classes $\boldsymbol{TC} = \{T_1, T_2, \ldots, T_n\}$. Instances of a particular transaction class, $T_i \in \boldsymbol{TC}$, use a subset of the buffer pools, say $\boldsymbol{BP}_i = \{B_1, B_2, \ldots B_b\}$. The elements of $\boldsymbol{BP}_i$ are determined by the set of database objects that are used by instances of T_i. The average number of logical reads per instance of T_i on buffer pool $B_j \in \boldsymbol{BP}_i$ is represented as $L_i(B_j)$.

We assume that the average response time for a transaction class T_i is directly proportional to the average data access time for instances of the class. The data access time for a transaction depends upon the number of *logical reads* issued by that transaction. So an estimate of the average response time per instance of transaction class T_i is given by

$$C_i = \sum_{j=1}^{b} L_i\left(B_j\right) \times costLR_j(m)$$

where $costLR_j(m)$ is the average cost of a logical read from buffer pool B_j of size m pages. We observed in our experiments that, as expected, many of the cost components of a logical read depend upon the size of the buffer pool in use.

We estimate the cost (response time) of a logical read on buffer pool j with m memory pages as follows:

$$costLR_j(m) = cpuLR + costAR_j(m) + costAW_j(m) + \left((1 - pAR_j(m)) \times miss_j(m) \times costPR_j\right) + \left((1 - pAW_j(m)) \times pD_j(m) \times costPW_j\right)$$

The cost of a logical read ($costLR_j(m)$), as indicated by the equation, contains several components. The first component is the processing cost associated with a logical read (*cpuLR*). In this paper we assume that the processing cost is not significant and can be set to zero.

The second component of the cost of a logical read is the delay added by I/O servers performing asynchronous reads ($costAR_j(m)$). We estimate the impact of the I/O servers by amortizing the cost of all asynchronous reads to a buffer pool across all logical reads ($noLR_j$) as follows:

$$costAR_j(m) = \frac{pAR_j(m) \times noPR_j(m) \times costPR_j}{noLR_j}$$

The cost of a physical read from buffer pool *j* ($costPR_j$) is estimated as the average of the physical read costs calculated at the initial data collection points. The number of asynchronous reads ($noAR_j(m)$) is dependent upon the buffer pool size and is calculated as a portion of the total number of physical reads ($noPR_j(m)$). The proportion of asynchronous reads to buffer pool *j* at memory size *m* is

$$pAR_j(m) = \frac{noAR_j(m)}{noPR_j(m)}$$

and the ratio is approximated at all values of *m* by a first-order polynomial and least squares approximation. The number of physical reads of buffer pool *j* at memory size *m* ($noPR_j(m)$) is approximated by

$$noPR_j(m) = miss_j(m) \times noLR_j$$

where the miss rate for buffer pool *j* at memory size *m* is

$$miss_j(m) = \frac{noPR_j(m)}{noLR_j}$$

The miss rate is approximated at all values of *m* by a second-order polynomial and least squares approximation. We first used Belady's equation [1] to approximate the hit rate but we found that our current method, provided that there are at least three data points, gives better approximations to miss rate curves in a wide variety of circumstances.

The third component of the cost of a logical read is the delay caused by I/O cleaners performing asynchronous writes. As with the I/O servers, we estimate the impact of the I/O cleaners by amortizing the cost of all asynchronous writes across all logical reads as follows:

$$costAW_j(m) = \frac{pAW_j(m) \times noPW_j(m) \times costPW_j}{noLR_j}$$

The cost of a physical write from buffer pool *j* ($costPW_j$) is estimated as the average of the physical write costs calculated at the initial data collection points. The num-

ber of physical writes of buffer pool *j* at all memory sizes *m* ($noPW_j(m)$) is approximated by a second order polynomial and least squares estimation. The number of asynchronous writes is dependent upon the buffer pool size and is calculated as a portion of the total number of physical writes. The proportion of asynchronous writes to buffer pool *j* at memory size *m* is

$$pAW_j(m) = \frac{noAW_j(m)}{noPW_j(m)}$$

and the ratio is approximated at all values of *m* by a first-order polynomial and least squares approximation.

The fourth component of the cost of a logical read, which is given by the factor

$$\left(1 - pAR_j(m)\right) \times miss_j(m) \times costPR$$

is the percentage of logical reads that result in a physical read. This percentage is determined by the miss rate of the buffer pool. The I/O servers also affect the miss rate of the buffer pool since they prefetch pages into the buffer pool.

The fifth component of the cost of a logical read, which is given by the factor

$$\left(1 - pAR_j(m)\right) \times miss_j(m) \times \left(\left(1 - pAW_j(m)\right) \times pD_j(m) \times costPW_j\right)$$

is the percentage of all logical reads that involve a physical write. In these cases, there are no clean buffer pages available for replacement so a dirty page must be written to disk before the new page can be read.

The possibility of having to write a dirty page from buffer pool *j* of size *m* ($pD_j(m)$) is the ratio of the number of synchronous writes on buffer pool j of size m to the number of physical reads on buffer pool j of size m given by:

$$pD_j(m) = \frac{noSW_j(m)}{noPR_j(m)}$$

This ratio is approximated at all values of *m* by a first-order polynomial and least squares approximation. I/O cleaners increase the probability that a free page is found by asynchronously writing dirty pages back to disk, which is captured by the factor $(1 - pAW_j(m))$ in the equation.

4. Experiments

We now discuss the results of a set of experiments we conducted to evaluate the performance of DRF. The experiments were run using DB2/UDB Version 5.2 under Windows NT on an IBM Netfinity 5000. The machine was configured with one 400 MHz processor, 1 GB of RAM and four 8.47 GB SCSI disks. DB2/UDB was configured with a disk page size of 8K bytes, four I//O Cleaners and two I/O servers. A reallocation unit of 10 pages was used in DRF.

The database schema and transaction workload are from the TPC-C benchmark [7]. The schema is composed of nine relations. The experimental database is approximately 300 MBs. It consists of 3 warehouses where each warehouse services

10 districts and each district has 3000 customers. Each warehouse stocks 100,000 items. The TPC-C workload consists of five transaction classes: Stock Level, Order Status, New Order, Payment and Delivery. The relative frequencies of instances of each class are specified in the benchmark.

Each experiment was run once while no other users were active on the machine and the workload was run against the system for ten minutes. We allow the application to run for 6 minutes in order to stabilize performance and then take the average of the performance statistics over the next 4 minutes All DB2/UDB performance measures are collected using the system's monitoring API [5]. We arrived at estimates of the number of logical reads to each buffer pool by transactions of each class by first independently running each class.

We evaluate the performance of DRF according to two criteria:

1. *The number of reallocation steps required to meet a goal.* We define the number of reallocation steps to be the number of different memory allocations that must be used before a goal is achieved. Brown [3] refers to this measure as the number of required adjustments to a performance knob. The number of reallocation steps used is an indication of the *effectiveness* of the algorithm, specifically a smaller number of steps indicates a more effective algorithm. It is also a measure of the *stability* of the algorithm, that is a stable algorithm requires a consistent number of steps across a variety of conditions. As explained in the previous section, DRF performs best if three initial data collection points are available for the curve-fitting component. We include these points in our count of reallocation steps in each experiment. We note, however, that this is pessimistic and in many realistic cases the old collection points can be reused.
2. *The percentages difference between the goal and real average response times.* The difference between the goal average response time and the real average response time achieved with the configuration suggested by DRF is a measure of the *accuracy* of the algorithm.

We do not report the computation times for DRF in the following discussions. We found that, in all cases, the computation time is not significant. The time will vary depending on the initial conditions. For the experiments reported here the computation time for DRF was in the range 0.5 to 0.9 seconds.

The experiments indicate the performance of DRF with an OLTP workload, typified by the TPC-C benchmark, under a variety of conditions. DB2/UDB was configured with 8 MB of memory (1000 pages) allocated to three buffer pools, which we identify as BP_D1, BP_D2 and BP_X. The database objects were assigned to buffer pools as follows:

- Warehouse and District tables to BP_D1;
- all other tables to BP_D2;
- all indexes to BP_X.

Each database object was placed on a single disk and all four disks were used to store the database.

Initial BP Config	Goal (sec)	Real (sec)	% Diff	No. Steps	Final BP Config (BP_D1, BP_D2, BP_X)
skewed	0.100	0.103	3.0	5	100, 330, 570
	0.090	0.089	1.1	4	50, 400, 550
	0.080	0.081	1.3	5	130, 300, 570
uniform	0.080	0.086	7.5	4	284, 383, 333
	0.075	0.073	2.7	4	234, 433, 333
	0.070	0.066	5.7	4	184, 483, 333

Table 1 : Experiment 1 – Cases 1 and 2

The following four sets of initial conditions are used in this set of experiments:

Case 1 - One target class and a skewed initial buffer pool allocation: The Delivery transaction class is not meeting its goal and BP_D1, BP_D2 and BP_X are assigned 250, 200 and 550 pages, respectively.

Case 2 - One target class and a uniform initial buffer pool allocation: The Delivery transaction class is not meeting its goal and BP_D1, BP_D2 and BP_X are assigned 334, 333 and 333 pages, respectively.

Case 3 - Two target classes and a skewed initial buffer pool allocation: The Delivery and New Order transaction classes are not meeting their goals and BP_D1, BP_D2 and BP_X are assigned 250, 200 and 550 pages, respectively.

Case 4 - Two target classes and a uniform initial buffer pool allocation: The Delivery and New Order transaction classes are not meeting their goals and BP_D1, BP_D2 and BP_X are assigned 334, 333 and 333 pages, respectively.

The results of the first two sets of experiments, where there is one class not meeting its goal, are shown in Table 1. In all cases, DRF converges to a reallocation of pages such that the Delivery transaction's real average response time is within 8% of goal. In some cases the real response time is below the goal and in some cases it is above. Two of the experiments require 5 reallocation steps and the rest require 4 reallocation steps. As we mentioned earlier, we are adding the three collection points to all experiments. If we used existing data points then one or two steps would be required to get an appropriate buffer pool allocation.

The results of the last two sets of experiments, where there are two classes not meeting their goals, are shown in Table 2. In all cases DRF converges to allocations such that both transactions' real average response times are within 16% of their goals and in 9 of the cases the real response times are within 10% of the goal.

Initial BP Config	New Order			Delivery			No Steps	Final BP Config
	Goal (sec)	Real (sec)	% Diff	Goal (sec)	Real (sec)	% Diff		BP_D1,BP_D2, BP_X
skewed	0.050	0.043	14	0.090	0.085	5.5	4	80, 350, 560
	0.045	0.045	0	0.100	0.096	4	4	100, 300, 600
	0.040	0.044	10	0.080	0.086	7.5	4	100, 340, 560
uniform	0.050	0.042	16	0.080	0.090	11.9	4	294, 373, 333
	0.045	0.042	6.7	0.075	0.075	0	4	274, 393, 333
	0.040	0.040	0	0.070	0.068	2.8	4	264, 403, 333

Table 2: Experiment 1 – Cases 3 and 4

We observe, based on the results of all four cases, that the initial buffer pool allocation does not have a negative impact on the performance of DRF. Table 2, however, would seem to indicate that the number of transaction classes not meeting their goals does have a negative impact on DRF's accuracy. It is more likely to be satisfied with an allocation that leaves the transactions slightly farther away from their goals.

One of the difficult problems we faced in running the experiments was choosing reasonable average response time goals, which leads us to wonder about the feasibility of requiring users to provide such goals. There is too much variability in real systems for average response time goals. A more flexible type of goal is required if goal-oriented approaches are to be practical. Percentile-type goals, such as 90% of Delivery transactions have a response time less than 0.1 seconds, are another alternative that may provide the necessary flexibility. Another approach to solving the problem is to let the system decide what is the best performance for a transaction class but allow the user to override the system in some situations.

5. Summary

In this paper, we described a self-tuning algorithm for the buffer area, which we call Dynamic Reconfiguration or DRF. It is a general algorithm that can be used with any relational DBMS that uses multiple buffer pools.

DRF improves upon previous self-tuning algorithms for the buffer area in several ways:

- DRF uses a more sophisticated response time estimator that accounts for the effects of dirty buffer pages and asynchronous reads and writes performed by system processes.
- DRF accounts for classes that share data pages. The dynamic tuning and fragment fencing algorithms do not consider shared data pages.
- DRF uses a data-oriented model of buffer organization. Previous algorithms all use a transaction class-oriented model.

We conclude, based on the results of our experiments, that DRF is effective, stable and accurate in the majority of cases. In the majority of cases DRF converged to actual response times that were within 10% of the goals. The experiments demonstrate the usefulness of goal-oriented resource management, generally, and dynamic buffer management, specifically, in a realistic DBMS environment.

In the future we plan to look at a number of issues leading from the work presented here. First, we plan to extend DRF to work with OLAP workloads, specifically TPC-H and TPC-R [9]. Second, we are looking at techniques for discovering and exploiting relationships among the buffer pools that can improve performance. Third, we are extending DRF so that it can balance memory allocations between the buffer pools and the sort area. Finally, we plan to look at a more proactive approach to dynamic resource management which uses information from the query optimizer to anticipate the resource needs of upcoming transactions.

References

1. L. Belady. A Study of Replacement Algorithms for a Virtual-Storage Computer, *IBM Systems Journal 5(2)*, July 1966.
2. K. Brown, M. Carey and M. Livny. Managing Memory to Meet Multiclass Workload Response Time Goals, *Proc. of 19th Int. Conf. on Very Large Databases*, Dublin, August 1993, pp. 328 - 341.
3. K. Brown, M. Carey and M. Livny. Goal-Oriented Buffer Management Revisited, *Proc. of the 1996 ACM SIGMOD Int. Conf. on Management of Data, Montreal*, June 1996, pp. 353 - 364.
4. J.-Y. Chung, D. Ferguson, G. Wang, C. Nikolaou and J. Teng. Goal-oriented dynamic buffer pool management for data base systems, *Proc. of Int. Conf. on Engineering of Complex Systems (ICECCS'95)*, November 1995.
5. IBM. *DB2 Universal Database*, http://www.software.ibm.com/data/db2/udb.
6. E. Isaacson and H. Keller. *Analysis of Numerical Methods*, John Wiley and Sons Inc., New York, 1966.
7. S.T. Leutenegger and D. Dias. A Modeling Study of the TPC-C Benchmark, *Proc. of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington D.C., 1993, pp.22 - 31.
8. C. Nikolaou, D. Ferguson and P. Constantopoulos. Towards Goal-Oriented Resource Management, *IBM Research Report* RC17019, April 1992.
9. Transaction Processing Performance Council, http://www.tpc.org.

Assigning Tasks to Resource Pools: A Fuzzy Set Approach

[1]A. de Korvin[1], S. Hashemi[1], G. Quirchmayr[2], R. Kleyle[3]

[1] University of Houston - Downtown
Houston Texas, 77002, USA

[2] Universität Wien
Institut für Informatik und Wirtschaftsinformatik
Liebiggasse 4, A-1010 Wien, Austria

[3] Indiana University - Purdue University at Indianapolis
Indianapolis, IN 46202, USA

Abstract. In this paper we address the problem of assigning tasks to resource pools. Each task has certain resource requirements, but with the capacity to provide these resources varying from pool to pool. We represent each task as a finite fuzzy set whose support consists of the resources and whose memberships reflect the degree of importance of each resource in performing this specific task. The resource pools are also represented by finite fuzzy sets having the same support, but now the memberships reflect the capability of a specific pool to provide each of these resources. We next define a measure of compatibility between each task and each resource pool. This compatibility is itself a fuzzy set which we defuzzify via the center of area (COA) method. We then develop an algorithm that describes how to recursively assign tasks to resource pools until some prespecified compatibility criterion has been violated. Finally, we add an assessment of cost to our algorithm, thereby enhancing its potential for practical application.

[1] This research was sponsored in part by ARO grant number DAAH-0495-1-0250 and NSF grant number CDA-9522157.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 102-114, 2000.

Introduction

Optimal job scheduling and resource allocation have recently been topics of interest in management science and operations research. Some of the early research in this area is summarized in the Sivazlian (1975) and in Arrow and Hurwicz (1977). For more recent work in this area, refer to Sengupta (1985) and Salmon (1991). Even more recently, fuzzy logic and fuzzy set theory have been used to model the uncertainty involved in such allocations. The authors of this work, de Korvin et al (1999), have extend to the fuzzy domain the algorithms of Carlier (1982) and Adams et al (1988) for optimal job scheduling to avoid bottlenecks. For more background on fuzzy optimization, refer to Delgado et al, (1994) and Verdegay, (1995). We also refer the reader to the works of Slany (1996) and Raggl and Slany (1998).

In this paper we address the problem of assigning tasks to resource pools. Each task has certain resource requirements, and each resource pool contains the same resources, but with the capacity to provide these resources varying from pool to pool. We represent each task as a finite fuzzy set whose support consists of the resources and whose memberships reflect the degree of importance of each resource in performing this specific task. If, for example, a task does not require a particular resource, the membership assigned to this resource is zero. On the other hand, a resource that is vital to the performance of this task will be assigned a high membership, perhaps even a membership of one (i.e. certainty). The resource pools are also represented by finite fuzzy sets whose supports again consist of the resources, but now the memberships reflect the capability of a specific pool to provide each of these resources. For example, if a specific resource pool cannot provide some resource, the membership associated with this resource is zero, whereas if a resource pool is extremely capable of providing a resource, a high membership will be assigned to this particular resource.

Following the approach initially given by Zadeh (1978) and expanded upon by Dubois and Prade (1980), we define a measure of compatibility between each task and each resource pool. This compatibility is itself a fuzzy set which we defuzzify via the center of area (COA) method that we briefly describe later. We next develop an algorithm that describes how to recursively assign tasks to resource pools until some pre-specified compatibility criterion has been violated. After this happens, no further tasks can be assigned until one or more tasks have been completed and, consequently, the demand on the resource pools is temporarily diminished.

Finally, we add an assessment of cost to our algorithm to enhance its potential for application in practical situations.

Motivation

A large task is to be accomplished. To carry out this task resources are needed. We have a set of resource pools. Every pool has the same set of resources, however the capabilities of these resources varies from pool to pool. For example, every pool may contain a printer and/or a programmer but the capabilities of the printer and/or programmer may be different in distinct pools. We would like to assign pools of resources to our task in some optimal or near optimal way. It should be noted that once a pool has been assigned, the remaining needed resources and their efficiency requirements must be updated as the requirements of the task change after resources are assigned. Another factor to be taken into consideration is the cost of pool usage. In what follows we develop a recursive algorithm to carry out such assignments. We view a task as consisting of needed resources together with required efficiency. It is therefore natural to represent a task as a fuzzy set of resources. Similarly, we view a pool of resources as a set of resources with specified efficiency, so again we represent a pool as a fuzzy set of resources. We need to match these sets and take cost into consideration. In addition, we would like to have some flexibility in our initial budget so if the need is great enough, we may to some extent go over our initial budget.

Defining the Problem

Consider a process P that consists of a set of tasks $\{T_1, T_2, \ldots\ldots, T_N,\}$. We assume that each task T_i, $(1 \le i \le N)$ requires resources $r_1, r_2, \ldots\ldots, r_n$, with performances $\alpha_{i1}, \alpha_{i2}, \ldots\ldots, \alpha_{in}$. That is, task T_i needs resource r_j, and on a scale from 0 to 1, we would like r_j to perform with efficiency α_{ij}, where $0 \le \alpha_{ij} \le 1$ for all indices $1 \le j \le 1$. We also have several resource pools labeled $RP_1, RP_2, \ldots\ldots, RP_m$. Each of these resource pools contains the same set of resources, $r_1, r_2, \ldots\ldots, r_n$, but with a performance index unique to the specific pool. For example, resource pool RP_k, will have performance indices $\beta_{k1}, \beta_{k2}, \ldots\ldots, \beta_{kn}$ corresponding to resources $r_1, r_2, \ldots\ldots, r_n$ respectively.

Tasks may be performed sequentially or in parallel. It is advantageous to have tasks performed in parallel, but if task T_i is assigned to resource pool RP_k, each resource in RP_k losses some capacity if called upon to handle an additional task. The resources could be hardware, e.g., CPU, memory, etc., or human resources, e.g., C^{++} programmers, JAVA programmers, etc. Obviously, if a CPU is assigned to task T_1, it will not perform as well if also assigned to task T_2 so that it must perform both tasks in parallel. When this happens, a deterioration of capacity takes place. Let d_{ikj} $(0 \le d_{ikj} \le 1)$ denote the deterioration that takes place when resource r_j in resource pool RP_k is assigned to task T_i. Suppose that initially the efficiency of r_j in pool RP_k was β_{kj}. Then, after the assignment of this resource in RP_k to task T_i, its efficiency deteriorates to $d_{ikj}\beta_{kj}$. If T_i is a difficult task or if r_j is already in use, $d_{ikj} < 1$, perhaps considerably less than one.

Typically we will assume that $m \leq N$, i.e., there are at least as many tasks as resource pools, although this assumption is not absolutely necessary. We also decide not to assign a task to a resource pool if the matching between needed and available resources is "too low", i.e., below a certain threshold. This matching, which is called compatibility, is defined below.

Compatibility

Mathematically, we represent both a task and a resource pool as fuzzy sets of resources. That is,

$$T_i = \sum_{k=1}^{n} \alpha_{ik} / r_k \qquad RP_t = \sum_{k=1}^{n} \beta_{tk} / r_k \ ,$$

where $r_1, r_2, \ldots\ldots, r_n$ denote the support of both fuzzy sets, and $\alpha_{i1}, \alpha_{i2}, \ldots\ldots, \alpha_{in}$ and $\beta_{k1}, \beta_{k2}, \ldots\ldots, \beta_{kn}$ (both defined above) constitute the respective memberships.

We define the compatibility of task T_i with resource pool RP_t as follows:

$$Comp[T_i, RP_t](u) = Sup\ T_i\ (r),$$

where the supremum over all resources is r such that $RP_t(r) = u$. This definition is due to Zadeh, (1978). To see how this definition works, let us consider the following example.

Let $T = .2 / r_1 + .6 / r_2 + .8 / r_3 \qquad RP = .8 / r_1 + .5 / r_2 + .2 / r_3$.

Then $Comp[T, RP](.2) = .8$, $Comp[T, RP](.5) = .6$, $Comp[T, RP](.8) = .2$.

Thus, we see that $Comp[T_i, RP_t]$ are fuzzy subsets of the unit interval [0, 1]. In the above example, using standard notation,

$$Comp[T, RP] = .8 /.2 + .6 /.5 + .2 /.8 .$$

In this example, the compatibility is mainly thought to be .2 (belief .8), but there is belief .6 that the compatibility might be.5, and a small belief (.2) that the compatibility is as high as .8. For a further development of the concept of compatibility, refer to Zadeh (1978), and Dubois and Prade (1980). Compatibility relations are discussed in Klir and Yuan (1995).

We now need to assign some tasks, i.e., certain subsets of $T_1, T_2, \ldots\ldots, T_N$, to the set of resource pools, $RP_1, RP_2, \ldots\ldots, RP_m$, and to compute the degree of fitness of such an assignment. Suppose, for example, that $m = 2$ and $N = 3$. If we assign T_1 and T_2 to RP_1 and RP_2 respectively, we have compatibilities $Comp[T_1, RP_1]$ and $Comp[T_2, RP_2]$ as previously defined. There are several ways in which we could define the compatibility of the assignment. We could, for example, average the two

compatibilities or simply take the intersection of the two fuzzy sets. We chose the intersection approach so that the compatibility of the above assignment becomes

$$\text{Comp}[T_1, RP_1] \wedge \text{Comp}[T_2, RP_2].$$

Thus, if

$$\text{Comp}[T_1, RP_1]\,](u_k) = \alpha_{k11}, \quad \text{Comp}[T_2, RP_2]\,](u_k) = \alpha_{k22},$$

then

$$(\text{Comp}[T_1, RP_1] \wedge \text{Comp}[T_2, RP_2])(u_k) = \alpha_{k11} \wedge \alpha_{k22},$$

where $\alpha_{k11} \wedge \alpha_{k22}$ denotes the smallest of the two elements α_{k11} and α_{k22}. Similarly, if we denote the assignment

$$\text{Comp}[T_i, RP_j]\,](u_k) = \alpha_{kij},$$

and if we denote the assignment of $T_{i_1}, T_{i_2}, ..., T_{i_p}$ to $RP_{m_1}, RP_{m_2}, ..., RP_{m_p}$ by $\text{Comp}[i_1, i_2, \ldots\ldots, i_p, m_1, m_2, \ldots\ldots, m_p,]$, then

$$\text{Comp}[i_1, i_2, \ldots\ldots, i_p, m_1, m_2, \ldots\ldots, m_p,](u_k) = \alpha_{ki_1m_1} \wedge \alpha_{ki_2m_2} \wedge ..... \wedge \alpha_{ki_pm_p}\ .$$

Thus we have extended the concept of the compatibility of a task with a resource pool to the concept of compatibility of the assignment of tasks $T_{i_1}, T_{i_2}, ..., T_{ip}$ to resource pools $RP_{m_1}, RP_{m_2}, ..., RP_{m_p}$.

Comparing Assignments

Each assignment generates a compatibility expressed as an intersection of the capabilities of an individual task to a research pool. These intersections are fuzzy subsets of the unit interval [0, 1]. An obvious way to compare such sets is to defuzzify them by the center of area (COA) method, and then compare the defuzzifications. In general, if C is a fuzzy subset of [0, 1], the defuzzification of C is given by

$$g_C = \frac{\sum_u uC(u)}{\sum_u C(u)}$$

if the support of C is countable, and

$$g_C = \frac{\int_0^1 uC(u)du}{\int_0^1 C(u)du}$$

if the support of C is a continuum.

Again suppose that m = 2 and N = 3, and suppose p = 2. We now consider all possible assignments of tasks to resource pools. That is, we must consider

C_1 = Comp[1,2; 1,2] C_2 = Comp[1,3; 1,2] C_3 = Comp[2,1; 1,2]
C_4 = Comp[2,3; 1,2] C_2 = Comp[3,1; 1,2] C_3 = Comp[3,2; 1,2]
C_7 = Comp[1,2; 1,1] C_8 = Comp[1,3; 1,1] C_3 = Comp[2,3; 1,1]
C_{10} = Comp[1,2; 2,2] C_{11} = Comp[1,3; 2,2] C_{12} = Comp[2,3; 2,2]

We then compute the defuzzifications g_k corresponding to C_k for $1 \le k \le 12$. If, for example, $g_2 > g_k$ for all $k \neq 2$, then since C_2 measures the compatibility of the assignment
$T_1 \rightarrow RP_1$ and $T_3 \rightarrow RP_2$, we start by assigning T_1 to and T_3 to RP_2.

There could be situations in which we may not opt to select the assignment of tasks to resource pools by defuzzifying by the COA method described above. The maximizing set approach proposed by Jain (1976) may also be used if we strike a balance between the compatibilities themselves and their degrees of belief (i.e., memberships). We illustrate the use of this approach in an example.

The Maximizing Set Approach

Suppose that

$$C_1 = .8/.9 + .8/.3 \qquad C_2 = .7/.6 + .5/.5.$$

In this example, the total support of the two fuzzy sets is {.3, .5, .6. .9}.

We then create the fuzzy maximizing set M in which the support is the set given above and the memberships are obtained by dividing each value in the support by the maximum value, which in this case is .9. Thus

$$M = \left(\frac{3}{9}\right)/.3 + \left(\frac{5}{9}\right)/.5 + \left(\frac{6}{9}\right)/.6 + \left(\frac{9}{9}\right)/.9 = .33/.3 + .56/.5 + .67/.6 + 1/.9$$

.

We next compute

$$M \wedge C_1 = .33/.3 + .8/.9 \qquad M \wedge C_2 = .5/.5 + .67/.6 .$$

Finally, we look for the highest membership in both $M \wedge C_1$ and $M \wedge C_2$. In this case it is .8 which is a membership in set $M \wedge C_1$. Thus, fuzzy set C_1 is considered to be "larger" than fuzzy set C_2.

Note: In this example, $g_{C_1} = 0.60$ and $g_{C_2} = 0.61$. Thus, has the COA method been used, C_2 would have been considered to be "larger" than C_1.

In summary, the assignment of tasks to resource pools may be considered to determine which assignment should be favored. Depending on the situation, we may use defuzzification via the COA method or the Jain's maximizing set method (if we want to favor assignments, which generate an especially high belief in a large compatibility.)

Updating the Resource Pools

Using the notation of the previous section, assume that the assignment [1,3; 1,2] has been made. That is, task T_1 has been assigned to resource pool RP_1, and task T_3 has been assigned to resource pool RP_2. Prior to the selection, assume that RP_1 and RP_2 are defined by

$$RP_1 = \sum_k \beta_{1k} / r_k \qquad RP_2 = \sum_k \beta_{2k} / r_k \ .$$

After the assignment,

$$RP_1 = \sum_k d_{11k}\beta_{1k} / r_k \quad RP_2 = \sum_k d_{32k}\beta_{2k} / r_k \ .$$

The d factors define the deterioration of the resources in pools RP_1 and RP_2.

We must now make the following decision:

Should task T_2 be run concurrently, or should we wait for one or more resource pool to be free? One way to approach this situation is to decide if the new, reduced capabilities of RP_1 and RP_2 are still high enough to run T_2 concurrently. In general, we have resource pools RP_1,RP_2, ….., RP_m, and we must decide if we can run some task T_i given the current load on these resource pools.

Let $\delta > 0$ be a threshold level, and consider the assignment [i_1, i_2, …. i_p, m_1, m_2, ..., m_p,]. Let T_{i^*} be an as yet unassigned task. We decide that T_{i^*} may potentially be assigned to RP_{m^*} if the center of mass g_c of Comp [i_1, i_2, …. i_p, m_1, m_2, ..., m_p] is at least δ. Of course there may be more than one resource pool for which the above assignment compatibility condition holds. This leads to the following assignment algorithm.

Assignment Algorithm

1. The assignment of tasks $T_{i_1}, T_{i_2}, \ldots\ldots, T_{i_p}$ has been made to resource pools RP_{m_1}, RP_{m_2}, ….., RP_{m_p}. Comp [i_1, i_2, …. i_p; m_1, m_2, ..., m_p,] has therefore been determined and fuzzy resource pools RP_{m_1}, RP_{m_2}, ….., RP_{m_p} have been updated by the appropriate deterioration factors as illustrated above.

2. Consider the set I = {T_i : i ∉ {i_1, i_2, ……., i_p}} of idle tasks. For each i ∈ I, compute C_{iw} = Comp [i_1, i_2, …. i_p, i; m_1, m_2, ..., m_p, m_w], $1 \le m_w \le m$.

3. Consider the set of C_{iw}, i ∈ I, and pick the "largest" of these, call it $C_{i^*w^*}$, using either the COA method or Jain's maximizing set to make this determination. Then replace the current assignment by [i_1, i_2, …. i_p, i*; m_1, m_2, ..., m_p, m_{w^*}].

4. If C[i_1, i_2, …. i_p, i*; m_1, m_2, ..., m_p, m_{w^*}] has COA $g_C < \delta$, stop and consider the assignment [i_1, i_2, …. i_p; m_1, m_2, ..., m_p] to be final. Tasks in idle set I must wait for some resources to become free. If $g_C \ge \delta$, go to 5.

5. Replace the set {i_1, i_2, ……., i_p} with { i_1, i_2, …. i_p, i*}, and go to 1.

Thus, the task that maximizes the capability of the assignment is chosen until the resources are overloaded enough so that the compatibility of the new assignment under consideration falls below some specified threshold δ.

Cost Considerations

Compatibility alone may not necessarily always determine the assignment of tasks to resource pools. Some cost is usually associated with tying up a resource pool. In what follows we assume that this cost can be translated into dollars. Moreover, this translation into dollars is not necessarily precise. Thus, the dollar amount that we are willing to spend is represented by fuzzy sets whose membership functions are trapezoids as illustrated in Figure 1 and Figure 2.

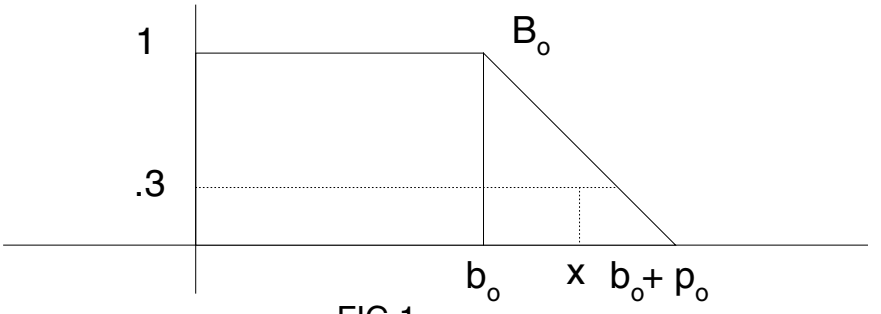


FIG 1

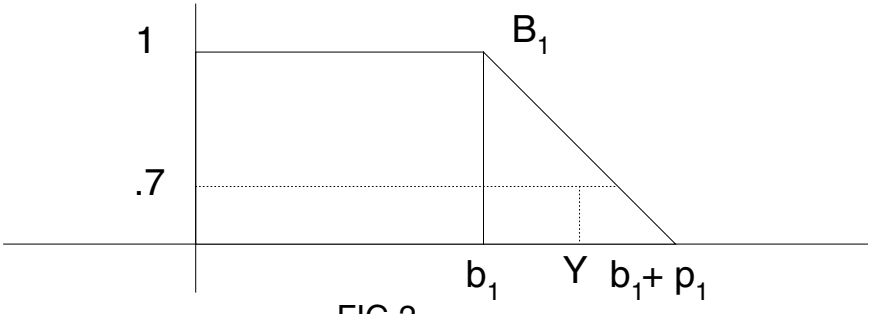


FIG 2

Figure 1 illustrates our initial amount, B_0. We are certain (belief 1) that we can spend up to b_0 dollars, we believe with strength .3 that we can spent up to x dollars, and we know that we cannot spend more than $b_0 + p_0$ under any circumstances. Figure 2 represents B_1, the amount remaining to be spent after the first assignment has been selected. For example, we believe with strength .7 that we still have y dollars to spend, we are certain that we have at least b_1 and at most $b_1 + p_1$ dollars to spend. Of course $b_1 < b_0$, and $b_1 + p_1 < b_0 + p_0$. Thus, $B_1(x) \leq B_0(x)$ for all x values.

Now in order to make an assignment of tasks to resource pools, we want to take into consideration both (1) compatibility and (2) cost. We now define the degree to which fuzzy sets B_0 and B_1 are equal. This measure is the fuzzy set whose membership function is

$$d[B_0 = B_1](x) = 1 - B_0(x) + B_1(x).$$

Note that since $B_1(x) \leq B_0(x)$, $0 \leq d[B_0 = B_1](x) \leq 1$ for all x. This definition is in the same spirit, but is somewhat simpler than, the one given by de Korvin et al (1994).

Motivation for this definition of degree of equality is given in Pedrycz and Gomide (1998).

In this particular application, $B_1(x) \le B_0(x)$ for all x. If this had not been the case, the membership function for fuzzy set $d[B_0 = B_1]$ becomes

$$d[B_0 = B_1](x) = 1 - \max\{B_0(x), B_1(x)\} + \min\{B_0(x), B_1(x)\}.$$

Now from figures 1 and 2 we see that

$$B_0(x) = \begin{cases} 1 & x \le b_0 \\ \dfrac{b_0 + p_0 - x}{p_0} & b_0 \le x \le b_0 + p_0 \\ 0 & x \ge b_0 + p_0 \end{cases},$$

and

$$B_1(x) = \begin{cases} 1 & x \le b_1 \\ \dfrac{b_1 + p_1 - x}{p_1} & b_1 \le x \le b_1 + p_1 \\ 0 & x \ge b_1 + p_1 \end{cases}.$$

Using the above algebraic representations of the trapezoidal membership functions, and assuming that $b_1 < b_0 < b_1 + p_1 < b_0 + p_0$, it can be shown that

$$d[B_0 = B_1](x) = \begin{cases} 1 & x \le b_1 \\ \dfrac{b_1 + p_1 - x}{p_1} & b_1 \le x \le b_0 \\ 1 - \dfrac{p_0(x - b_1) - p_1(x - b_0)}{p_0 p_1} & b_0 \le x \le b_1 + p_1 \\ 1 - \dfrac{b_0 + p_0 - x}{p_0} & b_1 + p_1 \le x \le b_0 + p_0 \\ 1 & x \ge b_0 + p_0 \end{cases}.$$

Now if

$$g(x) = 1 - \frac{p_0(x - b_1) - p_1(x - b_0)}{p_0 p_1}, \quad \text{for } b_0 \le x \le b_1 + p_1,$$

then

$$g'(x) = -\frac{p_0 - p_1}{p_0 p_1} = \frac{p_1 - p_0}{p_0 p_1} \qquad \text{for } b_0 \le x \le b_1 + p_1.$$

Therefore, if $p_0 < p_1$, $g'(x) > 0$ for $b_0 \le x \le b_1 + p_1$, and

$$\min_{b_0 \le x \le b_1 + p_1} g(x) = g(b_0) = \frac{p_1 - (b_0 - b_1)}{p_1},$$

whereas if $p_0 > p_1$, $g'(x) < 0$ for $b_0 \le x \le b_1 + p_1$, and

$$\min_{b_0 \le x \le b_1 + p_1} g(x) = g(b_1 + p_1) = \frac{p_1 - (b_0 - b_1)}{p_0}.$$

Consequently,

$$\inf_x d[B_0 = B_1](x) = \begin{cases} \dfrac{p_1 - (b_0 - b_1)}{p_1} & p_0 \le p_1 \\ \dfrac{p_1 - (b_0 - b_1)}{p_0} & p_0 \ge p_1 \end{cases}.$$

Now let

$$P_A = \inf_x d[B_0 = B_1](x),$$

where A denotes the assignment that reduced our initial capital B_0 to B_1. P_A, of course, is some number on unit interval [0, 1]. Let α, $0 \le \alpha < 1$, denote the weight that we assign to cost. We agree to select that assignment A which maximizes

$$(1 - \alpha)g_A + \alpha P_A,$$

where g_A is the defuzzified compatibility of assignment A. The previous algorithm remains in effect except that our selection criterion is now to maximize the above quantity instead of simply the compatibility. If $\alpha = 0$, cost is ignored, and we revert to the assignment algorithm as previously written. However, as α increases toward 1, cost becomes an increasingly important factor.

Conclusion

The algorithm developed in the present work allows for a systematic assignment of pools of resources to a given task. The pools consist of the same resources but the efficiency of the resources varies from pool to pool. The task at hand has certain priorities assigned to the needed resources. The algorithm performs a compatibility match between the resources and the priorities of the task. After each assignment, the priorities are updated. The cost of using a pool is factored into the decision. The available budget has some build in flexibility and, if the need is large enough, allows to go over the initial budget, to some degree. Compatibility and cost may be weighted, reflecting their importance in the decision making process.

References

Adams, J., Balas, E. and Zawack, D. (1988): The shifting bottleneck procedure for job scheduling. *Management Science* **34**, 391-401.

Arrow, K. and Hurwicz, L., (eds.), (1977): *Studies in the Resource Allocation Process*. Cambridge, UK, Cambridge University Press.

Carlier, J. (1982): The one-machine sequencing problem. *European Journal of Operations Research* **11**, 42-47.

de Korvin, A. Bourgeois, B. and Kleyle, R. (1994), Extracting fuzzy rules under uncertainty and measuring definablity using rough sets. *Journal of Intelligent and Fuzzy Systems* **2**, 75-87.

de Korvin, A., Kleyle, R., Hashemi, S., and Quirchmayr, G. (1999), Assignment of tasks to competing nodes when task duration times are fuzzy. To appear in the Journal of *Intelligent and Fuzzy Systems*.

Delgado, , M., Verdegay, J. and Vila, M., (1994) . *Fuzzy Optimization: Recent Advances*. Heidelberg, Germany: Physica-Verlag.

Dubois, D. and Prade, H., (1980), *Fuzzy Sets and Systems: Theory and Applications*. New York: Academic Press.

Jain, R. (1976), Decision making in the presence of fuzzy variables. *IEEE Transactions on Systems, Man and Cybernetics* SMC-**6**, 698-703.

Klir, G. and Yuan, B., (1995), *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ, Prentice Hall.

Raggl, A., Slany, W.(1998), A reusable iterative optimization library to solve combinatorial problems with approximate reasoning. *International Journal of Approximate Reasoning* 19 (1-2) 161-191.

Salmon, M. (1991), *Deterministic Lotsizing Models for Production Planning*. New York.

Sengupta, J. (1985), *Optimal Decisions Under Uncertainty: Models, Methods and Management*. New York, Springer-Verlag.

Sivazlian, B., (1975), *Optimization Techniques in Operations Research*. Englewood Cliffs, NJ, Prentice-Hall.

Slany, W. (1996), Scheduling as a fuzzy multiple criteria optimization problem. *Fuzzy Sets and Systems* 78, 197-222.

Verdegay, J, (1995), Fuzzy optimization: models, methods and perspectives. *Sixth IFSA World Congress*, July 22, 1995, vol. **1**, 21-28,; Sao Paulo, Brazil.

Yager, R. 1988. On ordered weighted averaging aggregation operations in multicriteria decision making. IEE Transactions on Systems, Man and Cybernetics, 18: 183-190.

Zadeh, L., (1978), Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* **1**, 3 - 28.

Protocol for Taking Object-Based Checkpoints

Katsuya Tanaka and Makoto Takizawa

Dept. of Computers and Systems Engineering
Tokyo Denki University
Email {katsu, taki}@takilab.k.dendai.ac.jp

Abstract. Object-based checkpoints are consistent in the object-based system but may be inconsistent according to the traditional message-based definition. We present a protocol for taking object-based checkpoints among objects. An object to take a checkpoint in the traditional message-based protocol does not take a checkpoint if the current checkpoint is object-based consistent with the other objects. The number of checkpoints can be reduced by the object-based protocol.

1 Introduction

Distributed applications are composed of multiple objects. An object is an encapsulation of data and methods for manipulating the data. A method is invoked by a message passing mechanism. A conflicting relation among the methods is defined based on the semantics of the object [4]. If a pair of methods op_1 and op_2 conflict, a state of the object obtained by performing op_1 and op_2 depends on the computation order of op_1 and op_2.

A state of an object is saved in the *log* at a checkpoint. A faulty object o is *rolled back* to the checkpoint and then is restarted. Here, objects which have received messages sent by objects rolled back also have to be rolled back. Papers [1, 2, 7, 9–11, 14] discuss how to take a globally consistent checkpoint of multiple objects. In the object-based systems, types of messages, i.e. *request* and *response* messages are exchanged among objects. Since the traditional checkpoints are defined in terms of messages exchanged among objects, the definition is referred to as *message-based*. We newly define *object-based consistent* (*O-consistent*) checkpoints which can be taken based on conflicting relations among methods in various types of invocations like synchronous and asynchronous ones. The O-consistent checkpoint may be inconsistent with the traditional message-based definition. In this paper, we present a protocol where O-consistent checkpoints are taken for objects without suspending the computation of methods. By taking only the O-consistent checkpoints, the number of checkpoints can be reduced.

In section 2, we discuss the object-based checkpoints. In sections 3 and 4, we show a checkpointing protocol and restarting protocol, respectively.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 115–124, 2000.

2 Object-Based Checkpoints

2.1 Objects

A distributed system is composed of multiple objects $o_1, \ldots, o_n$. Each object o_i is an encapsulation of data and a set of methods. In this paper, we assume methods are synchronously or asynchronously invoked by using the remote procedure call. On receipt of a *request op*, *op* is performed on the object o_i. Here, let op^i denote an instance of *op*. Then, a *response* message is sent back. *op* may furthermore invoke another method op_1, i.e. *nested* invocation. If op_1 is synchronously invoked, *op* blocks until receiving the response of op_1. In the asynchronously invocation, *op* is being performed without blocking. A message m *participates* in a method *op* if m is a request or response of *op*. Let $Op(m)$ denote a method in which a message m participates.

Let $op(s)$ denote a state obtained by performing a method *op* on a state s of an object o_i. $op_1 \circ op_2$ shows that a method op_2 is performed after op_1 completes. op_1 and op_2 of an object o are *compatible* iff $op_1 \circ op_2(s) = op_2 \circ op_1(s)$ for every state s of o [4]. Otherwise, op_1 and op_2 *conflict*. An object supports two kinds of methods, i.e. *update* method which changes the state of the object and *non-update* one. The types of methods are assumed to be specified with the conflicting relation in the definition of the object.

2.2 Object-based checkpoints

A local checkpoint c^i for an object o_i is taken where a state of o_i is stored in the log l_i. If o_i is faulty, o_i is rolled back to c^i by restoring the state stored in the log l_i. Then, other objects have to be rolled back to the checkpoints if they had received messages sent by o_i. A *global checkpoint* c is a tuple $\langle c^1, \ldots, c^n \rangle$ of the local checkpoints. From here, a term *checkpoint* means a *global* one.

Suppose an instance op_1^i invokes a method op_2 in o_j. Figure 1 shows possible checkpoints for o_i and o_j. Here, c_3^i is not taken if op_2^j is synchronously invoked. Let $\pi_j(op^j, c^j)$ be a set of instances performed on o_j, which precede op^j and succeed c^j or are being performed at c^j in o_j. For example, $\pi_j(op_2^j, c_1^j) = \{op_{21}^j, \ldots, op_{2l}^j\}$ in Figure 1.

We discuss whether or not each checkpoint $\langle c_k^i, c_h^j \rangle$ can be taken in the object-based system. For example, $\langle c_1^i, c_3^j \rangle$ is message-based inconsistent in Figure 1 because a message m_1 is an orphan. If op_2^j is non-update, the state denoted by c_2^j is the same as c_3^j and c_4^j. That is, $\langle c_1^i, c_3^j \rangle$ and $\langle c_1^i, c_4^j \rangle$ show the same state as $\langle c_1^i, c_2^j \rangle$. $\langle c_1^i, c_2^j \rangle$ is message-based consistent. Hence, o_j can be restarted from any of c_3^j and c_4^j if o_j can be restarted from c_2^j. Here, $\langle c_1^i, c_3^j \rangle$ is consistent in the object-based system (O-consistent). $\langle c_1^i, c_4^j \rangle$ is also O-consistent. A local checkpoint c^i is *complete* if there is no method being performed at c^i. For example, c_3^i is

incomplete in Figure 1. Table 1 summarizes the message-based inconsistent but O-consistent checkpoints, where checkpoints marked * are incomplete if op_2^j is being performed.

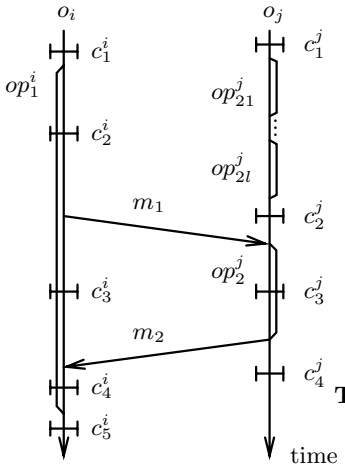


Fig. 1. Possible checkpoints.

o_i	o_j	Conditions
c_1^i	c_3^j*, c_4^j	op_2^j is a non-update type.
c_2^i	c_3^j*, c_4^j	op_2^j is a non-update type.
c_4^i, c_5^i	c_1^j	op_2^j is a non-update type and no method in $\pi_j(op_2^j, c_1^j)$ conflicts with op_2^j.
	c_2^j, c_3^j*	op_2^j is a non-update type.

Table 1. O-consistent checkpoints for Figure 2.

[**Definition**] A message m is *influential* iff a method instance op_2^j of an object o_j sends a message m to o_i and one of the following conditions is satisfied:

1. op_1^i is an update type if m is a request message, i.e. op_2^j invokes op_1^i in o_i.
2. If m is a response of op_2^j, op_2^j is an update type or conflicts with some instance in $\pi_j(op_2^j, c)$ where c is a local checkpoint most recently taken in o_j. □

If op^i is aborted, only instances receiving influential messages from op^i are required to be aborted. In Figure 1, suppose op_1^i sends an asynchronous update request m_1. Here, m_1 is influential from the definition. If o_i is rolled back to c_2^i, o_j is also rolled back.

[**Definition**] A global checkpoint $c = \langle c^1, \ldots, c^n \rangle$ is *object-based consistent* (*O-consistent*) iff there is no influential orphan message at c. □

3 Checkpointing Protocol

3.1 Communication-induced protocol

We briefly present a basic communication-induced checkpointing protocol where objects are not suspended while checkpoints are being taken. First, each object o_i initially takes a local checkpoint c_0^i. An initial checkpoint $\langle c_0^1, \ldots, c_0^n \rangle$ is assumed to be consistent. After sending and receiving messages, a first local checkpoint c_1^i is taken for o_i. Thus, the tth local checkpoint c_t^i is taken after c_{t-1}^i ($t > 0$). Here, t denotes a *checkpoint identifier* of c_t^i.

Suppose a local checkpoint c_t^i is taken for an object o_i after c_{t-1}^i. Then, only if o_i sends a message m to another object o_j, m is marked *checkpointed*. By sending m, o_i notifies the destination objects that o_i has taken c_t^i. Thus, o_i does not send any additional control message to take local checkpoints. Here, suppose c_{u-1}^j is taken for o_j and a checkpoint $\langle c_{t-1}^i, c_{u-1}^j \rangle$ is consistent. On receipt of the checkpointed message m from o_i, a local checkpoint c_u^j is taken for o_j at which o_j saves a state which is most recent before o_j receives m. The state saved here is referred to as *checkpoint* state. In fact, a current state and the operation $rec(m)$ for receiving m are stored in the log l_j. A compensating operation $\sim rec(m)$ to remove every effect done by $rec(m)$ is assumed to be supported for every object. If o_j is rolled back to c_u^j, the state saved in the log is first restored, and then $\sim rec(m)$ is performed.

In the object-based system, o_j does not take c_u^j if $\langle c_t^i, c_{u-1}^j \rangle$ is O-consistent. We discuss how o_j decides if $\langle c_t^i, c_{u-1}^j \rangle$ is O-consistent.

3.2 O-consistent checkpoints

A vector of checkpoint identifiers $\langle cp_1, \ldots, cp_n \rangle$ is manipulated for an object o_i to identify the tth local checkpoint c_t^i of o_i. Each cp_k is initially 0. Each time a local checkpoint is taken for o_i, $cp_i := cp_i + 1$. A message m which o_i sends to o_j after taking $c_{cp_i}^i$ carries a vector $m.cp = \langle m.cp_1, \ldots, m.cp_n \rangle$, where $m.cp_k$ is cp_k of o_i $(k = 1, \ldots, n)$.

On receipt of a message m from o_j, $cp_j := m.cp_j$ in o_i. cp_i shows a checkpoint identifier which o_i has most recently taken. Another variable cp_h shows a newest checkpoint identifier of an object o_h which o_i knows $(h = 1, \ldots, n, j \neq i)$. That is, $\langle c_{cp_1}^i, \ldots, c_{cp_n}^i \rangle$ shows a current checkpoint which o_i knows. If $m.cp_j > cp_j$ in o_i, o_i finds that o_j has taken c_u^j following $c_{cp_j}^i$ where $u = m.cp_j$. A local checkpoint c_t^i is identified by a vector $\langle c_t^i.cp_1, \cdots, c_t^i.cp_n \rangle$ where each $c_t^i.cp_j$ shows a value of cp_j when c_t^i is taken for o_i.

A local checkpoint c_t^i has a bitmap $c_t^i.BM = b_1 \cdots b_n$ where each hth bit b_h is used for an object o_h $(h = 1, \ldots, n)$. Suppose c_t^i is taken for o_i. Here, $c_t^i.b_i = 1$ and $c_t^i.b_j = 0$ for $j = 1, \ldots, n$, $j \neq i$. If $c_t^i.b_j = 0$ and there is data to be sent to o_j, o_i sends a checkpointed message m with the data to o_j. Here, $m.BM := c_t^i.BM$.

On receipt of m from o_i, o_j takes a local checkpoint c_u^j. Here, $c_u^j.b_k := m.b_k$ $(k = 1, \ldots, n, k \neq j)$ and $c_u^j.b_j := 1$ while the checkpoint identifier vector is updated as presented here. Thus, "$c_t^i.b_k = 1$" shows that o_i knows o_k takes a local checkpoint by the checkpointing protocol initiated by a same object.

[Definition] c_t^i and c_u^j are in the *same generation* if $c_t^i.BM \cap c_u^j.BM \neq \phi$ and $c_t^i.cp_k = c_u^j.cp_k$ for every object o_k such that $c_t^i.b_k = c_u^j.b_k = 1$. □

Each time an object o_i sends a message m to o_j, a message *sequence number* sq and a *subsequence number* ssq_j are incremented by one $(j = 1, \ldots, n)$.

The sequence number $m.sq$ and a vector of the subsequence numbers $m.ssq = \langle m.ssq_1, \ldots, m.ssq_n \rangle$ are carried by m. Variables $rsq_1, \ldots, rsq_n$ and $rssq_1, \ldots, rssq_n$ are manipulated in o_j. On receipt of m from o_i, o_j accepts m if $m.ssq_j = rssq_i + 1$. That is, o_j delivers messages from each object in the sending order. Then, $rssq_i := rssq_i + 1$ and $rsq_i := m.sq$. $rssq_i$ and rsq_i show subsequence and sequence numbers of message which o_j has most recently received from o_i. m also carries a vector $m.rq = \langle m.rq_1, \ldots, m.rq_n \rangle$ where $m.rq_k = rsq_k$ ($k = 1, \ldots, n$). Here, $m.rq_k$ shows a sequence number of message which o_i has received from o_j just before c_t^i and $t = m.cp_i$ ($k = 1, \ldots, n$).

On receipt of a message m from o_i, o_j collects a set M_j of messages $m_{j1}, \ldots, m_{jl_j}$ which o_j has sent to o_i after c_{u-1}^j and o_i has received before c_t^i. Here, $m_{jh}.sq \leq m.rq_j$ [Figure 2]. Messages which o_j sends after c_{u-1}^j are stored in the sending log of o_j. Suppose o_j receives a checkpointed message m from o_i. If $m.cp_i > cp_i$, o_j knows o_i takes c_t^i. o_j collects every message m' which o_j has sent after c_{u-1}^j and $m'.sq < m.rq_j$ in the set M_j.

[Theorem] A message m_{jh} which o_j sends to o_h after taking a local checkpoint c_{u-1}^j before c_u^j is *influential* if m_{jh} is a request and $Op(m_{jh})$ is an update type, or m_{jh} is a response and $Op(m_{jh})$ is an update type or conflicts with some update method in $\pi_j(Op(m_{jh}), c_{u-1}^j)$. □

The condition of the theorem is referred to as *influential message* (*IM*) condition. Only if some message in M_j is decided to be influential by the *IM* condition, o_j takes a local checkpoint c_u^j.

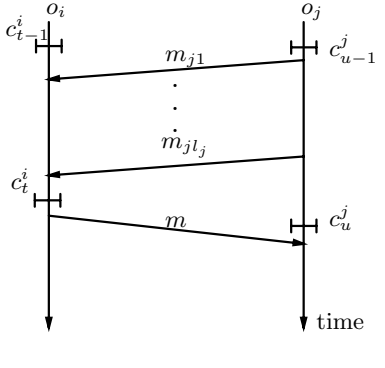


Fig. 2. Influential messages.

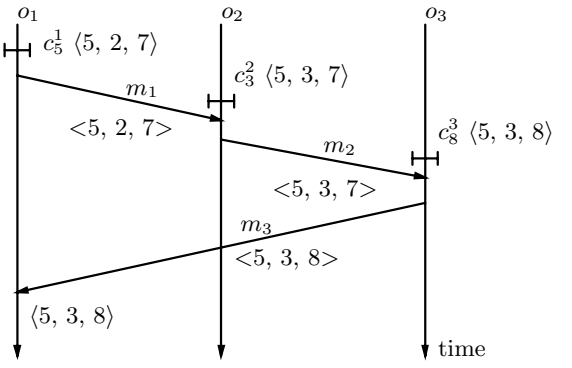


Fig. 3. Cyclic checkpointing.

3.3 Cyclic checkpointing

[**Example 1**] Suppose each of three objects o_1, o_2, and o_3 has initially checkpoint identifier vector $cp = \langle cp_1, cp_2, cp_3 \rangle = \langle 4, 2, 7 \rangle$ [Figure 3]. First, a local checkpoint c_5^1 is taken for o_1. Here, $cp = \langle 5, 2, 7 \rangle$. o_1 sends m_1 with $\langle 5, 2, 7 \rangle$ to o_2 after taking c_5^1. o_2 takes c_3^2 on receipt of m_1 where $c_3^2.cp = \langle 5, 3, 7 \rangle$. Then, o_2 sends

m_2 with $\langle 5, 3, 7\rangle$ to o_3. On receipt of m_2, o_3 takes c_8^3 and sends m_3 with $\langle 5, 3, 8\rangle$ to o_1. o_1 takes c_6^1. Then, o_2 and o_3 take new local checkpoints as presented here. Thus, the checkpointing procedure cannot be terminated in o_1, o_2, and o_3. This is *cyclic checkpointing.* □

Here, when o_1 receives m_3, o_1 is not required to take a local checkpoint because a checkpoint $\langle c_5^1, c_3^2, c_8^3\rangle$ taken already is consistent. A pair of checkpoints identified by $\langle 5, 2, 7\rangle$ and $\langle 5, 3, 8\rangle$ are in the same generation.

[**Example 2**] Here, let a notation "$\langle cp_1, \ldots, cp_n\rangle_{b_1 \ldots b_n}$" show $cp = \langle cp_1, \ldots, cp_n\rangle$ and $BM = b_1 \cdots b_n$. In Figure 3, o_1 sends o_2 a message m_1 with $\langle 5, 2, 7\rangle_{100}$, i.e. $cp = \langle 5, 2, 7\rangle$ and $BM = 100$ after c_5^1. On receipt of m_1, cp is changed to $\langle 5, 2, 7\rangle$ in o_2. Then, o_2 sends m_2 with $\langle 5, 3, 7\rangle_{110}$ to o_3 after c_3^2. c_8^3 is taken for o_3 and then sends m_3 with $\langle 5, 3, 8\rangle_{111}$ to o_1. On receipt of m_3, o_1 knows the checkpointing procedure has been initiated by o_1 because $\langle 5, 2, 7\rangle$ and $\langle 5, 3, 8\rangle$ are in the same generation. □

The checkpoint identifier vector $cp = \langle cp_1, \ldots, cp_n\rangle$ and the bitmap $BM = b_1 \cdots b_n$ are manipulated in o_i on receipt of m as follows:

- $cp_k := \max(cp_k, m.cp_k)$ if $m.b_k = 1$ for every k $(\neq i)$.
- $BM := BM \cup m.BM$.

The checkpoint identifier vector cp and the bitmap BM are saved in the checkpoint log $c_{cp_i}^i$ of o_i only if they are changed. In Figure 3, on receipt of m_3, $c_5^1.cp = \langle 5, 3, 8\rangle$. If $cp_1 > m.cp_1$, another object initiates the checkpointing procedure independently of o_1. A local checkpoint is taken for o_1 if there is some influential message in M_1.

3.4 Merge of checkpoints

[**Example 3**] In Figure 4, every object has a checkpoint identifier vector $cp = \langle 4, 3, 7, 2\rangle$. Suppose o_1 and o_4 independently take checkpoints. o_1 sends m_1 after c_5^1 with $\langle 5, 3, 7, 1\rangle_{1000}$, i.e. $cp = \langle 5, 3, 7, 1\rangle$ and $BM = 1000$. On receipt of m_1, o_2 takes c_4^2 and then sends m_2 with $\langle 5, 4, 7, 1\rangle_{1100}$. On the other hand, o_4 takes c_2^4 with $\langle 4, 3, 7, 2\rangle_{0001}$ and then sends m_4 to o_3. o_3 takes c_8^3 with $\langle 4, 3, 8, 2\rangle_{0011}$ and then sends m_3 to o_2. o_2 receives m_3 with $\langle 4, 3, 8, 2\rangle_{0011}$ from o_3 after c_4^2 with $cp = \langle 5, 4, 7, 1\rangle$. o_3 receives m_2 with $\langle 5, 4, 7, 1\rangle_{1100}$ after c_8^3 with $cp = \langle 4, 3, 8, 2\rangle$. One way is that o_2 and o_3 take c_5^2 with $\langle 4, 5, 8, 2\rangle_{0111}$ and c_9^3 with $\langle 5, 4, 9, 3\rangle_{1110}$, respectively. Here, $\langle c_5^1, c_4^2, c_9^3, c_3^4\rangle$ and $\langle c_6^1, c_5^2, c_8^3, c_2^4\rangle$ are taken for o_1, o_2, o_3, and o_4.

Suppose o_4 is faulty and is rolled back to c_3^4. Then, o_3 is rolled back to c_9^3 and then o_2 is rolled back to c_4^2. Here, o_3 is required to be furthermore rolled back to c_8^3 and o_3 is also rolled back to c_2^4. In the worst case, each object is rolled back to the local checkpoints n times for the number n of objects [6]. □

In order to prevent such a *cascading* rollback, we take an approach to merging multiple checkpoints to one. In Figure 4, o_2 receives m_3 after c_4^2. Here, $\langle c_5^1, c_4^2\rangle$

with $BM = 1100$ and $\langle c_8^3, c_2^4\rangle$ with $BM = 0011$ are merged into $\langle c_5^1, c_4^2, c_8^3, c_2^4\rangle$ with $BM = 1111$.

[**Merge of checkpoints**] After c_t^i, o_i receives a message m.

1. If a checkpoint c_u^i denoted by $m.cp$ is not in the same generation as c_t^i, i.e. $c_u^i.BM \cap m.BM \neq \phi$, $c_t^i.cp_k := m.cp_k$ if $c_t^i.b_k = 0$ and $m.b_k = 1$ for every k ($\neq i$), and $c_t^i.BM := c_t^i.BM \cup m.BM$.
2. Otherwise, $c_t^i.BM := c_t^i.BM \cup m.BM$ and $c_t^i.cp_k := \max(c_t^i.cp_k, m.cp_k)$ for every k ($\neq i$). □

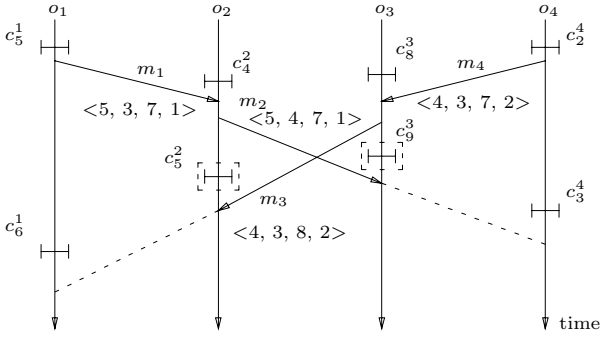


Fig. 4. Checkpoints.

[**Theorem**] A set of local checkpoints which belong to the same generation with the merge procedure is O-consistent. □

By the merging procedure, a new local checkpoint is not taken for o_2 even if o_2 receives messages after m_3 in Figure 4.

4 Rollback Recovery

4.1 Restarting protocol

If an object o_i is faulty, o_i is rolled back to the local checkpoint c_t^i. Other objects which have received influential messages sent by o_i after c_t^i are also required to be rolled back. Messages which o_i sends are recorded in the sending log. o_i has to send a rollback request message *R-Req* to every object o_j which o_i has sent influential messages after c_t^i. In order to decide to which objects *R-Req* is sent, o_i manipulates a log SL_t^i as follows:

- When a local checkpoint c_t^i is taken for o_i, SL_t^i is initiated to be empty.
- If o_i sends an influential message m to o_j, $SL_t^i = SL_t^i \cup \{o_j\}$.

If o_i is rolled back to c_t^i, o_i sends *R-Req* to every object o_j in SL_t^i. Here, *R-Req* contains the following information:

- A vector $cp = \langle cp_1, \ldots, cp_n\rangle$ of c_t^i to which o_i is rolled back.

- A bitmap $RB = rb_1 \ldots rb_n$ where each rb_k is 1 if o_i knows o_k is rolled back to a same generation checkpoint as c_t^i, otherwise, $rv_b = 0$.

Suppose an object o_i is faulty and is rolled back to c_t^i. o_i sends *R-Req* to every o_k in SL_u^j with $c_u^j.cp$ and RB where $rb_i := 1$. Then, o_i is suspended. On receipt of *R-Req* from o_i, o_j is also suspended. o_j discards *R-Req* if *R-Req*.rv_j $= 1$ since o_j has been already rolled back in this generation. Otherwise, $rb_j :=$ 1 and $RB := RB \vee$ *R-Req*.RB. o_j looks for an oldest local checkpoint c_u^j where $cp_i =$ *R-Req*.cp_i. If o_j finds c_u^j, c_u^j is a *rollback point* of o_j. Otherwise, the most recent checkpoint where $cp_i <$ *R-Req*.cp_i is a *rollback point*. Then, o_j collects a set RL^j of messages which o_j has received from o_i after c_u^j. If there is some influential message in RL^j, o_j is rolled back to the *rollback point* c_u^j. Then, o_j sends *R-Req* to every o_k in SL_u^j with RB and $c_u^j.cp$. If o_j received no influential message from o_i, o_j discards *R-Req* since o_j is not required to be rolled back. If o_j does not send *R-Req* to any objects, o_j sends the restart request message *Res-Req* to o_i. Otherwise, o_j waits for *Res-Req* from every object in SL_u^j. Then, o_j sends *Res-Req* to o_i.

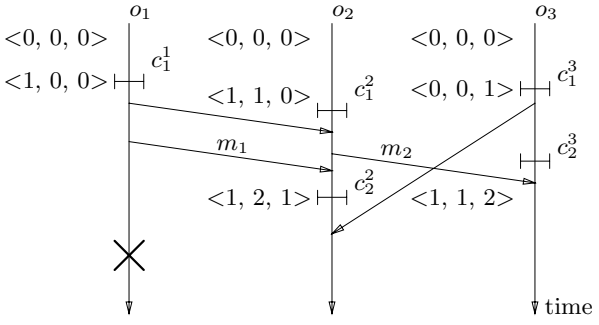


Fig. 5. Restarting procedure.

[**Example 4**] In Figure 5, if o_1 is faulty, o_1 is rolled back to c_1^1. o_1 is suspended and finds that o_1 has sent an influential message to o_2 by searching SL_1^1. Then, o_1 sends *R-Req* to o_2 with $cp = \langle 1, 0, 0 \rangle$ and $RB = 100$. On receipt of *R-Req* from o_1, o_2 finds an oldest local checkpoint c_1^2 where $cp_1 = 1$ because *R-Req*.cp_1 $= 1$. Since m_1 in RL^1 is influential, o_2 is rolled back to c_1^2. *R-Req*.$rb_2 = 1$. Then, o_2 sends *R-Req* to o_3 if m_2 is influential. On receipt of *R-Req* from o_2, o_3 is rolled back to c_2^3 if m_2 is influential. Otherwise, o_3 just discards *R-Req*, sends back the *Res-Req* to o_2, and then continues the computation. On receipt of *Res-Req* from o_3, o_2 sends *Res-Req* to o_1 and is restarted. □

4.2 Synchronous restarting protocol

In the protocol, each object is not required to be restarted simultaneously with other objects. This protocol is effective if only a few number of objects are rolled

back after some faulty object is rolled back. However, the more number of objects to be rolled back, the longer it takes to recover from the fault. In order to resolve the difficulty, we show a synchronous restarting protocol.

Suppose an object o_i is faulty and is rolled back to the local checkpoint c_t^i. o_i is suspended and broadcasts *R-Req* to all objects with $c_t^i.cp$. On receipt of *R-Req* from o_i, o_j is suspended. Then, the value $c_u^j.cp_i$ is compared with $R\text{-}Req.cp_i$ where c_u^j is a most recent local checkpoint. Suppose $R\text{-}Req.cp_i \geq c_u^j.cp_i$. Since o_j has not taken a same generation checkpoint with c_t^i, o_j is not rolled back. o_j sends back a message *no* to o_i and then is restarted. Otherwise, o_j sends *yes* to o_i. o_i finds a group of objects to be rolled back by using a bitmap $RB = rb_1 \ldots rb_n$. Each variable rb_k is initially 0 ($1 \geq k \geq n$). On receipt of *yes* from o_k, $rb_k := 1$. After receiving messages from all the objects, o_i sends *Rollback* with RB to o_k where $rb_k = 1$. On receipt of *Rollback* from o_i, o_j is rolled back to the *rollback point* if o_j had received any influential message from o_k where $rb_k = 1$. Then, o_j sends back *Done* to o_i. On receipt of *Done* from all the objects which o_i has sent *Rollback*, o_i sends *Res-Req* to the objects and then is restarted. On receipt of *Res-Req*, o_j is restarted.

[**Example 5**] Suppose there are four objects o_1, o_2, o_3, and o_4 as shown in Figure 6. Here, suppose o_1 is faulty and is rolled back to the checkpoint c_1^1. o_1 broadcasts *R-Req*. On receipt of *R-Req* from o_1, o_2 and o_3 send *yes* and o_4 sends *no* to o_1 since $c_1^1.cp_1 = c_1^2.cp_1 = c_1^3.cp_1$. On receipt of the messages, $rb = 1110$ in o_1. o_1 sends *Rollback* to o_2 and o_3. On receipt of *Rollback*, o_2 is rolled back to c_1^2 if m_1 is influential. Similarly, o_3 is rolled back to c_1^3 if m_2 is influential. □

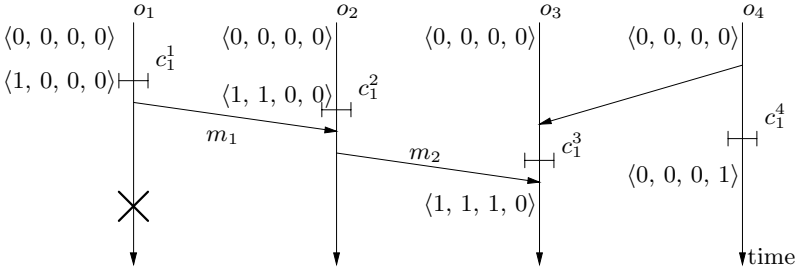


Fig. 6. Synchronous restarting procedure.

5 Concluding Remarks

We discussed how to take *object-based consistent* (*O-consistent*) checkpoints which may be inconsistent with the traditional message-based definition. We defined *influential messages* on the basis of the conflicting relation of requests where the methods are synchronously or asynchronously invoked in the nested manner. Only objects receiving influential messages are rolled back if the senders

of the influential messages are rolled back. At the *O-consistent checkpoint*, there is no orphan influential message. We presented the protocol for taking O-consistent checkpoints where no object is suspended in taking checkpoints. The number of local checkpoints can be reduced by the O-checkpoints. We presented the restarting protocol after some faulty object is rolled back.

References

1. Bhargava, B. and Lian, S. R., "Independent Checkpointing and Concurrent Rollback for Recovery in Distributed Systems — An Optimistic Approach," *Proc. of IEEE SRDS-7*, pp. 3-12, 1988.
2. Chandy, K. M. and Lamport, L., "Distributed Snapshots : Determining Global States of Distributed Systems," *ACM TOCS*, Vol. 3, No. 1, pp. 63–75, 1985.
3. Fischer, M. J., Griffeth, N. D., and Lynch, N. A., "Global States of a Distributed System," *IEEE Trans. on Software Engineering*, Vol. SE-8, No. 3, pp.198–202, 1982.
4. Garcia-Molina, H., "Using Semantics Knowledge for Transaction Processing in a Distributed Database," *Proc. of ACM SIGMOD*, Vol. 8, No. 2, pp. 188-213, 1983.
5. Helary, J.-M., Netzer, R. H.B., and Raynal, M., "Consistency Issues in Distributed Checkpoints," *IEEE Trans. on Software Engineering*, Vol. 25, No. 2, pp. 274–281, 1999.
6. Higaki, H., Sima, K., Tanaka, K., Tachikawa, T., and Takizawa, M., "Checkpoint and Rollback in Asynchronous Distributed Systems," *Proc. of IEEE INFOCOM'97*, pp. 1000–1007, 1997.
7. Koo, R. and Toueg, S., "Checkpointing and Rollback-Recovery for Distributed Systems," *IEEE TOCS*, Vol. C-13, No. 1, pp. 23–31, 1987.
8. Lin, L. and Ahamad, M., "Checkpointing and Rollback-Recovery in Distributed Object Based Systems," *Proc. of IEEE SRDS-9*, pp. 97–104, 1990.
9. Leong, H. V. and Agrawal, D., "Using Message Semantics to Reduce Rollback in Optimistic Message Logging Recovery Schemes," *Proc. of IEEE ICDCS-14*, pp.227–234, 1994.
10. Manivannan, D. and Singhal, M., "A Low-Overhead Recovery Technique Using Quasi-Synchronous Checkpointing," *Proc. of IEEE ICDCS-16*, pp.100–107, 1996.
11. Ramanathan, P. and Shin K. G., "Checkpointing and Rollback Recovery in a Distributed System Using Common Time Base," *Proc. of IEEE SRDS-7*, pp. 13–21, 1988.
12. Tanaka, K. and Takizawa, M., "Distributed Checkpointing Based on Influential Messages," *Proc. of IEEE ICPADS'96*, pp. 440–447, 1996.
13. Tanaka, K., Higaki, H., and Takizawa, M., "Object-Based Checkpoints in Distributed Systems," *Journal of Computer Systems Science and Engineering*, Vol. 13, No.3, pp. 125–131, 1998.
14. Wang, Y. M. and Fuchs, W. K., "Optimistic Message Logging for Independent Checkpointing in Message-Passing Systems," *Proc. of IEEE SRDS-11*, pp. 147-154, 1992.

This article was processed using the LaTeX macro package with LLNCS style

From Object-Oriented to Aspect-Oriented Databases

Awais Rashid[1], Elke Pulvermueller[2]

[1]Computing Department, Lancaster University, Lancaster LA1 4YR, UK
marash@comp.lancs.ac.uk
[2]Wilhelm Schickard Institute for Computer Science, University of Tuebingen, 72076 Tuebingen, Germany
pulvermueller@acm.org

Abstract. Over the recent years aspect-oriented programming (AOP) has found increasing interest among researchers in software engineering. *Aspects* are abstractions which capture and localise cross-cutting concerns. Although persistence has been considered as an aspect of a system aspects in the persistence domain in general and in databases in particular have been largely ignored. This paper brings the notion of aspects to object-oriented databases. Some cross-cutting concerns are identified and addressed using aspects. An aspect-oriented extension of an OODB is discussed and various open issues pointed out.

1 Introduction

Over the recent years aspect-oriented programming (AOP) [18, 19] has found increasing interest among researchers in software engineering. It aims at easing software development by providing further support for modularisation. *Aspects* are abstractions which serve to localise any cross-cutting concerns e.g. code which cannot be encapsulated within one class but is tangled over many classes. A few examples of aspects are memory management, failure handling, communication, real-time constraints, resource sharing, performance optimisation, debugging and synchronisation. In AOP classes are designed and coded separately from aspects encapsulating the cross-cutting code. An *aspect weaver* is used to merge the classes and the aspects.

Aspect-orientation appears to be following the same development phases as object-orientation. Introduced through object-oriented programming in the late 1960s (SIMULA-67) object-orientation is now employed in a wide range of software development activities such as analysis, design, modelling, etc. It has also been successfully applied in the areas of databases and knowledge-bases. Likewise, research in aspect-orientation is now progressing from programming into other areas such as specification [6, 7, 9, 30] and design [15, 17, 41]. Its applicability in the area of databases has, however, not yet been explored. Although persistence has been considered as an aspect of a system [27, 41] aspects in the persistence domain in general and in databases in particular have been largely ignored. This paper brings the notion of aspects to object-oriented databases in order to achieve a better separation of concerns. Reflecting on the fact that AOP is not limited to object-oriented programming languages [19], we are of the view that aspects can be employed in database technology other than OODBs e.g. relational, object-relational, active

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 125–134, 2000.

databases, etc. This will, however, form the subject of a future paper. The discussion in this paper focuses on extending object-oriented databases with aspects.
The next section provides an overview of aspect-oriented programming. Section 3 identifies some of the cross-cutting concerns in OODBs and discusses how aspects can be employed to address these effectively. This is followed by a description of an aspect-oriented extension of an object-oriented database. Extending object-oriented databases with aspects is a new concept and a number of issues need to be resolved before it can be considered as mature database technology. Some of the key open issues are identified in section 5. Section 6 discusses some related work while section 7 summarises and concludes the paper.

2 Aspect-Oriented Programming

Aspect-oriented programming aims at achieving a better separation of concerns by localising cross-cutting features e.g. code implementing non-functional requirements such as memory management, failure handling, debugging, synchronisation, etc. Code for debugging purposes or for object synchronisation, for instance, is spread across all the classes whose instances have to be debugged or synchronised, respectively. Although patterns [12] can help to deal with such cross-cutting code by providing guidelines for a good structure, they are not available or suitable for all cases and mostly provide only partial solutions to the code tangling problem. With AOP, such cross-cutting code is encapsulated into separate constructs: the aspects. As shown in fig. 1 classes are designed and coded separately from code that cross-cuts them. The links between classes and aspects are expressed by explicit or implicit *join points*. [17] categorises[1] these links as:

- *open:* both classes and aspects know about each other
- *class-directional:* the aspect knows about the class but not vice versa
- *aspect-directional:* the class knows about the aspect but not vice versa
- *closed:* neither the aspect nor the class knows about the other

An *aspect weaver* is responsible for merging the classes and the aspects with respect to the join points. This can be done statically as a phase at compile-time or dynamically at run-time [16, 19].
Different AOP techniques and research directions can be identified. They all share the common goal of providing an improved separation of concerns. AspectJ [3] is an aspect-oriented extension to Java. The environment offers an aspect language to formulate the aspect code separately from Java class code, a weaver and additional development support. AOP extensions to other languages have also been developed. [8] describes an aspect language and a weaver for Smalltalk. An aspect language for the domain of robust programming can be found in [11].
Other AOP approaches aiming at achieving a similar separation of concerns include subject-oriented programming [13], composition filters [1] and adaptive programming [23, 28]. In subject-oriented programming different subjective perspectives on a single object model are captured. Applications are composed of "subjects" (i.e. partial object models) by means of declarative composition rules. The composition filters approach extends an object with input and output filters. These filters are used to localise non-functional code. Adaptive programming is a special case of AOP where one of the building blocks is expressible in terms of graphs. The other building

[1] The categorisation determines the reusability of classes and aspects

blocks refer to the graphs using traversal strategies. A traversal strategy can be viewed as a partial specification of a class diagram. This traversal strategy cross-cuts the class graphs. Instead of hard-wiring structural knowledge paths within the classes, this knowledge is separated.
AOP also bears a close relation to generative programming [10] and reflection [41]. Experience reports and assessment of AOP can be found in [17, 31].

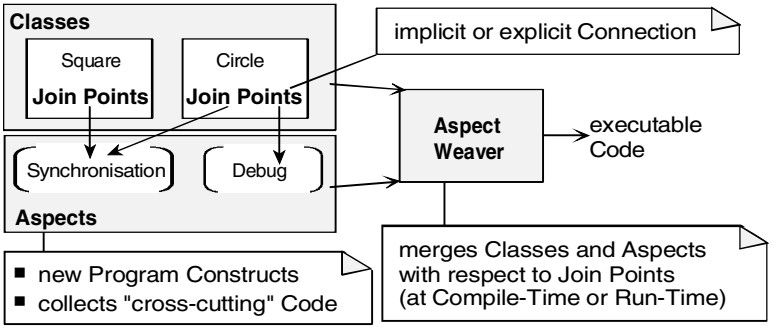


Fig. 1. Aspect Oriented Programming

3 Aspects in Object-Oriented Databases

In this section we identify some of the cross-cutting concerns in object databases and discuss how these can be addressed effectively using aspects. The first three examples are from the evolution domain. Our previous work on object database evolution [32, 33, 34, 35, 36, 37, 39] formed the motivation behind exploring the applicability of aspects in this area. This is followed by an example based on clustering. Some other cross-cutting concerns have also been pointed out. It should be noted that the following discussion regards aspects as persistent abstractions residing in the database. This is a natural extension of existing work on lifetime of aspects [16, 19, 26] which argue that at least some of the aspects should live for the lifetime of the program execution and not die at compile-time.

3.1 Instance Adaptation

Our first example is based on instance adaptation during class versioning [5, 29, 40]. Class versioning allows several versions of one type to be created during evolution. An instance is bound to a specific version of the type and when accessed using another type version (or a common type interface) is either converted or made to exhibit a compatible interface. This is essential to ensure structural consistency. A detailed description of class versioning is beyond the scope of this paper. Interested readers are referred to [5, 29, 40].
We first consider the instance adaptation strategy of ENCORE [40]. A similar approach is employed by AVANCE [5]. As shown in figure 2, applications access instances of a class through a *version set interface* which is the union of the properties and methods defined by all versions of the class. Error handlers are employed to trap incompatibilities between the version set interface and the interface of a particular class version. These handlers also ensure that objects associated with the class version exhibit the version set interface. As shown in fig. 2(b) if a new class version modifies the version set interface (e.g. if it introduces new properties and methods) handlers for the new properties and methods are introduced into all the former versions of the type.

On the other hand, if creation of a new class version does not modify the version set interface (e.g. if the version is introduced because properties and methods have been removed), handlers for the removed properties and methods are added to the newly created version (cf. fig. 2(c)).

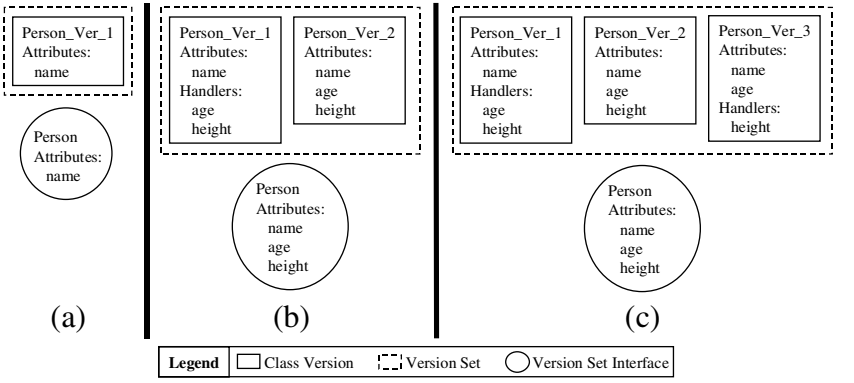


Fig. 2. Class Versioning in ENCORE

The introduction of error handlers in former class versions is a significant overhead especially when, over the lifetime of the database, a substantial number of class versions can exist prior to the creation of a new one. Besides, if the behaviour of some handlers needs to be changed maintenance has to be performed on all the class versions in which the handlers were introduced. To demonstrate our solution we have chosen the scenario in fig. 2(b). Similar solutions can be employed for other cases. As shown in fig. 3(a) instead of introducing the handlers into the former class versions they are encapsulated in an aspect. Links between aspects and class versions are *open* [17] as an aspect needs to know about the various class versions it can be applied to while a class version needs to be aware of the aspect that needs to be woven to exhibit a specific interface. Fig. 3(b) depicts the case when an application attempts to access the *age* and *height* attributes in an object associated with version 1 of *class Person*. The aspect containing the handlers is woven into the particular class version. The handlers then simulate (to the application) the presence of the missing attributes in the associated object.

Encapsulating handlers in an aspect offers an advantage in terms of maintenance as only one aspect is defined for a set of handlers for a number of older class versions. Behaviour of the handlers can be modified within the aspect instead of modifying them within each class version. Aspects also help separate the instance adaptation strategy from the class versions. For example, let us suppose one wants to employ a different instance adaptation approach, the use of update/backdate methods for dynamic instance conversion between class versions (as opposed to simulating a conversion) [29]. In this case only the aspects need to be modified without having the problem of updating the various class versions. These are automatically updated to use the new strategy when the aspect is woven. The aspect-oriented approach has a run-time overhead as aspects need to be woven and unwoven (if adaptation strategies are expected to change). However, this overhead is smaller than that of updating and maintaining a number of class versions. The overhead can be reduced by leaving an

aspect woven and only reweaving if the aspect has been modified. Details of the aspect-oriented instance adaptation approach have been reported in [38].

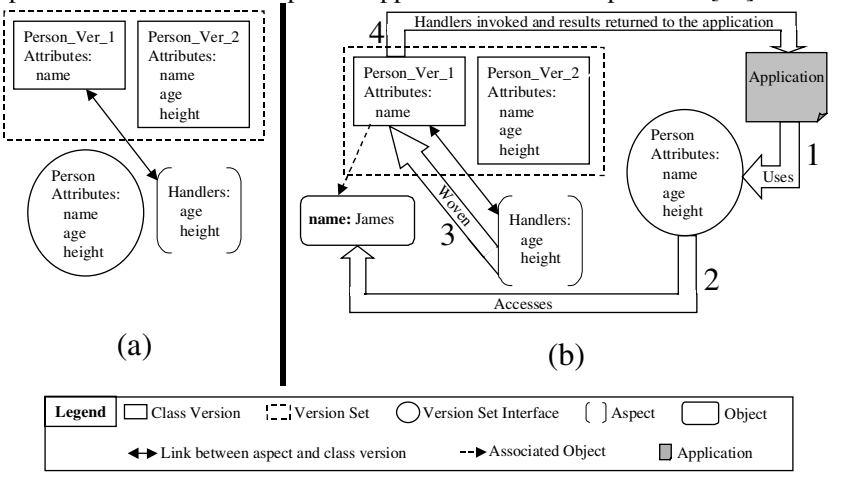


Fig. 3. The Aspect-Oriented Instance Adaptation Approach

3.2 Inheritance

The modification of the inheritance hierarchy in an object database requires rules to resolve conflicts between locally defined and inherited class members [4, 34]. If the system supports multiple inheritance additional rules need to be defined for duplication resolution due to the multiple-path problem [4, 34]. These rules are implemented in a separate system component which is delegated the responsibility of enforcing these rules when new classes are introduced or existing inheritance relationships modified. Alternatively, in a self-descriptive object-oriented model they are implemented in the system class used to instantiate the class meta-objects. In either case modifying the behaviour of the rules introduces an additional evolution problem as consequences for existing classes and their instances can be severe. Besides, there is an overhead in terms of checking for duplication resolution when the system supports multiple inheritance but the inheritance chain does not include any multiple inheritance links. These problems can be addressed by considering inheritance as an aspect of the system.

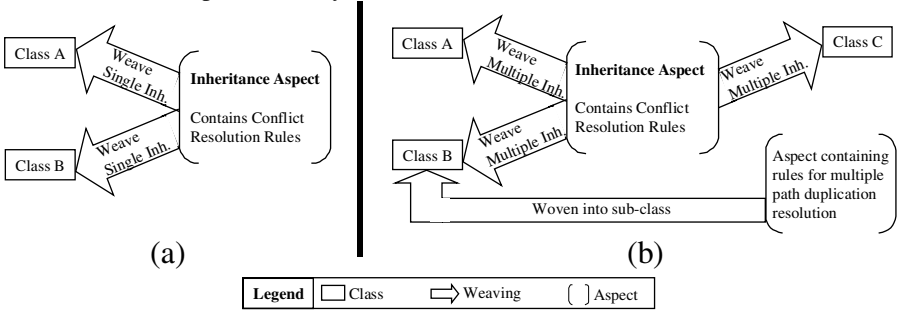


Fig. 4. Inheritance as an aspect of a system

As shown in fig. 4 inheritance relationships can be woven into classes using an *inheritance aspect*. Existing inheritance relationships can be unwoven and new ones woven in order to modify the inheritance hierarchy. Conflict resolution rules need to be woven or rewoven only if they have not previously been woven or have been modified respectively. It should be noted that the conflict resolution rules can be encapsulated in a separate aspect. For simplicity we have included them in the inheritance aspect. If the conflict resolution rules are modified only the conflict resolution aspect would need to be rewoven. Provided the system offers an evolution framework with a flexible instance adaptation strategy such as the one described in section 3.1 changes to instances of existing classes can be propagated relatively easily. As shown in fig. 4(b) duplication resolution rules to counter the multiple-path problem need to be introduced into the classes using multiple inheritance only[2]. This reduces the overhead of checking for duplication resolution when multiple inheritance is not being used. This also makes it possible to swiftly transform a single inheritance system to a multiple inheritance one and vice versa.

3.3 Versioning

Versioning of objects has always been a key requirement for databases especially systems aiming at providing integrated support for applications such as computer aided design (CAD), computer aided software engineering (CASE), etc. Existing work on versioning e.g. [4, 14] recognises that all objects residing in the database do not need to be versionable. If versioning features are encapsulated in an aspect these can be woven into those objects only that need to be versioned and not others. This also reduces the overhead to check whether an object is versioned or not. Besides, if the system supports class versioning, a generic versioning aspect could encapsulate the versioning functionality while specific aspects could provide additional object versioning and class versioning semantics to objects and classes respectively.

3.4 Clustering

Traditionally it is the task of the application programmer to ensure that related objects are clustered together. However, the task is easy only for a small number of objects. As the number of objects that need to be clustered increases (it should be noted that the clustering reasons could be different for different groups of objects) the programmer's task becomes more and more complicated. The situation worsens if the same object needs to be clustered together with more than one group. Considering clustering as an aspect of data residing in a database would allow managing these complex scenarios transparently of the programmer. The programmer can specify clustering as an aspect of a group of objects regardless of whether some objects in the group also form part of another group. When the clustering aspects are woven an efficient storage strategy can be worked out by the system[3]. Furthermore, if the clustering requirements for an object change the programmer can re-configure the clustering aspect to indicate which group of objects should be clustered with this object. This will help manage the physical reorganisation of the various clustered objects transparently of the programmer. It should also be noted that clustering is not necessarily an aspect of all the objects residing in the database. Introducing clustering as an aspect allows only those objects having this aspect to be clustered.

---

[2] Issues relating to weaving an aspect as a consequence of another aspect being woven have been discussed in [20]

[3] This can be done by the weaver.

3.5 Other possible aspects

Constraints can be considered as an aspect of the object database. Traditionally constraints are specified at the application level or through a DBMS service. Considering constraints an aspect of database entities and providing a concrete abstraction would simplify their specification and management. Access rights, security and data representation can also be regarded as aspects.

4 An Aspect-Oriented Extension of an OODB

In [36] we described a higher level model of an object-oriented database. The model is based on the observation that three types of entities exist within an object-oriented database. These are:

- Objects
- Meta-objects
- Meta-classes

As shown in fig. 5(a) entities of each type reside within a specific *virtual space* within the database. Objects reside in the *object space* and are instances of classes. Classes together with defining scopes, class properties, class methods, etc. form the *meta-object space* [35]. Meta-objects are instances of meta-classes which constitute the *meta-class space*.

For prototyping the aspects presented in section 3 we have extended the model with an *aspect-space* (fig. 5(b)). Aspects reside in the aspect space and are instances of the meta-class aspect. For simplicity we have shown only the versioning aspect.

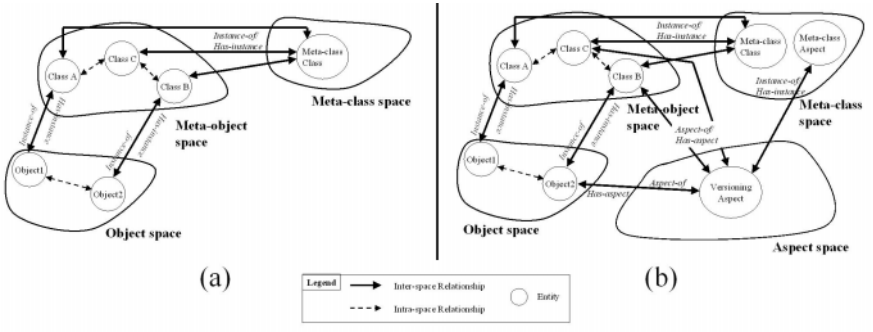


Fig. 5. Virtual Spaces in an Object Database

Fig. 5(b) also shows that the notion of inter-space[4] and intra-space relationships[5] introduced in [36] seamlessly extends to the aspect space. Aspects in the aspect space bear a*spect-of/has-aspect* relationships with entities residing in other virtual spaces. These relationships indicate an *open* link [17] between an aspect and the corresponding entity. *Class-directional* and *aspect-directional* links [17] can also be

[4] Inter-space relationships are those where participating entities reside in different virtual spaces e.g. relationships among meta-objects and objects.

[5] Intra-space relationships are those where the participating entities belong to the same virtual space e.g. relationships among objects, relationships among meta-objects.

created as the system supports uni-directional relationships [36]. Implementation of the semantics of *closed* links has not yet been explored. Although not shown in figure 5(b) intra-space relationships can exist within an aspect space. For example, a specific class versioning aspect can have a specialisation relationship with the versioning aspect. Note that the specialisation does not need to have semantics similar to the specialisation relationship (inheritance) in object-oriented models. It can simply be a conceptual specialisation.

As shown in figure 5(b), versioning is an aspect of Object 2, Class B and Class C. These are the entities that will be versioned. Class A and object 1 will not be versioned. It should be noted that object 1 could have a versioning aspect despite Class A not having one.

The aspect-oriented extension has been layered on top of the Jasmine object database management system[6]. Weaving is carried out by weavers developed for the specific purposes. Development of a more sophisticated weaver is in progress. It is worth mentioning that the aspect-oriented instance adaptation strategy has been incorporated into the SADES evolution system [32, 33, 34, 37, 39] and has proved to be a more flexible implementation as compared to instance adaptation in existing systems [38].

5 Open Issues

This paper introduces the idea of persistent aspects. One of the open research issues is the persistent representation of an aspect. This is because, besides aspects in the persistence domain described in this paper, application programs employing aspect-oriented programming techniques and run-time aspects will require aspects to outlive the program execution i.e. there will be a need to make aspects persistent. An example scenario is an automated software development environment where both components and aspects reside in a database. The appropriate components and aspects are retrieved by the assembling process which carries out the weaving. [31] also identifies the need of an aspect repository when employing aspects in a distributed environment. Due to the different aspect representations e.g. AspectJ, Composition Filters, etc. used in application programs, persistent representation of aspects needs careful exploration. Persistent aspects and dynamic weaving introduce additional overhead at run-time and can be feasible only with efficient weaving mechanisms. The development of efficient weavers is therefore an important research issue.

Other issues that need to be addressed include the applicability of the concepts in relational, object relational, active and deductive databases, etc. Another area worth exploring is whether querying support should be extended to aspects residing in the database. *Aspect-based querying and retrieval* can also pose interesting research questions.

6 Related Work

Although separation of concerns in object-oriented databases has not been explicitly considered, some of the existing work falls in this category. The concept of object version derivation graphs [25] separates version management from the objects. A similar approach is proposed by [32, 33, 34, 37, 39] where version derivation graphs manage both object and class versioning. Additional semantics for object and class versioning are provided separately from the general version management technique. [24] employs propagation patterns [21, 22] to exploit polymorphic reuse mechanisms

[6] http://www.cai.com/

in order to minimise the effort required to manually reprogram methods and queries due to schema modifications. Propagation patterns are behavioural abstractions of application programs and define patterns of operation propagation by reasoning about the behavioural dependencies among co-operating objects. [36] implements inheritance links between classes using semantic relationships which are first class-objects. The inheritance hierarchy can be changed by modifying the relationships instead of having to alter actual class definitions. In the *hologram approach* proposed by [2] an object is implemented by multiple instances representing its many faceted nature. These instances are linked together through aggregation links in a specialisation hierarchy. This makes objects dynamic since they can migrate between the classes of a hierarchy hence making schema changes more pertinent.

7 Conclusions

The use of AOP to achieve a better separation of concerns has shown promising results [17]. Although some of the existing work in object-oriented databases implicitly addresses cross-cutting concerns, no attempts have been made to capture these explicitly. The novelty of our work is in extending the notion of aspects to object-oriented databases in order to capture these concerns explicitly. We have identified a number of cross-cutting features and discussed how these can be effectively addressed using aspects. Some examples have been discussed in order to demonstrate the effectiveness of aspects in localising the impact of changes, hence making maintenance easier. We have presented an aspect-oriented extension of an object database which is used to prototype the various examples. The extension is natural and seamless and does not affect existing data or applications. Our work in the immediate future will focus on persistent representations of aspects and development of efficient weaving mechanisms. We believe that these issues will be crucial for the effective application of aspects in object databases. Development of an aspect-oriented evolution framework for object databases is another area of interest. We are also interested in exploring the applicability of aspects in database technology other than OODBs.

References

[1] Aksit, M., Tekinerdogan, B., "Aspect-Oriented Programming using Composition Filters", *Proceedings of the AOP Workshop at ECOOP '98, 1998*

[2] Al-Jadir, L., Leonard, M., "If We Refuse the Inheritance …", *Proceedings of DEXA 1999, LNCS 1677, pp. 560-572*

[3] AspectJ Home Page, http://aspectj.org/, *Xerox PARC, USA*

[4] Banerjee, J. et al., "Data Model Issues for Object-Oriented Applications", *ACM Transactions on Office Information Systems, Vol. 5, No. 1, Jan. 1987, pp. 3-26*

[5] Bjornerstedt, A., Hulten, C., "Version Control in an Object-Oriented Architecture", *In Object-Oriented Concepts, Databases, and Applications (eds: Kim, W., Lochovsky, F. H.), pp. 451-485*

[6] Blair, L., Blair, G. S., "The Impact of Aspect-Oriented Programming on Formal Methods", *Proceedings of the AOP Workshop at ECOOP '98, 1998*

[7] Blair, L., Blair, G. S., "A Tool Suite to Support Aspect-Oriented Specification", *Proceedings of the AOP Workshop at ECOOP '99, 1999*

[8] Boellert, K., "On Weaving Aspects", *Proc. of the AOP Workshop at ECOOP '99*

[9] Clarke, S., *et al.*, "Separating Concerns throughout the Development Lifecycle", *Proceedings of the AOP Workshop at ECOOP '99, 1999*

[10] Czarnecki, K., "Generative Programming: Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-based Component Models", *PhD Thesis, Technical University of Ilumenau, Germany, 1999*

[11] Fradet, P., Suedholt, M., "An Aspect Language for Robust Programming", *Proceedings of the AOP Workshop at ECOOP '99, 1999*

[12] Gamma, E. *et al.*, "Design Patterns - Elements of Reusable Object-Oriented Software", *Addison Wesley, c1995*

[13] Harrison, W., Ossher, H., "Subject-Oriented Programming (A Critique of Pure Objects)", *Proceedings of OOPSLA 1993, ACM SIGPLAN Notices, Vol. 28, No. 10, Oct. 1993, pp. 411-428*

[14] Katz, R. H., "Toward a Unified Framework for Version Modeling in Engineering Databases", *ACM Computing Surveys, Vol. 22, No. 4, Dec. 1990, pp. 375-408*

[15] Kendall, E.A., "Role Model Designs and Implementations with Aspect Oriented Programming", *Proceedings of OOPSLA 1999, ACM SIGPLAN Notices, Vol. 34, No. 10, Oct. 1999, pp. 353-369*

[16] Kenens, P., *et al.,* "An AOP Case with Static and Dynamic Aspects", *Proceedings of the AOP Workshop at ECOOP '98, 1998*

[17] Kersten, M. A., Murphy, G. C., "Atlas: A Case Study in Building a Web-based Learning Environment using Aspect-oriented Programming", *Proc. of OOPSLA 1999, ACM SIGPLAN Notices, Vol. 34, No. 10, Oct. 1999, pp. 340-352*

[18] Kiczales, G., Irwin, J., Lamping, J., Loingtier, J., Lopes, C., Maeda, C., Mendhekar, A., "Aspect-Oriented Programming", *ACM Computing Surveys, Vol. 28, No. 4, Dec. 1996*

[19] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., Irwin, J., "Aspect-Oriented Programming", *Proceedings of ECOOP '97, LNCS 1241, pp. 220-242*

[20] Klaeren, H., Pulvermueller, E., Rashid, A., Speck, A., "Supporting Composition using Assertions*, Computing Department, Lancaster University, Technical Report No: CSEG/4/00*

[21] Lieberherr, K. J., Huersch, W., Silva-Lepe, I., Xiao, C., "Experience with a Graph-Based Propagation Pattern Programming Tool", *Proc. of the International CASE Workshop, IEEE Computer Society 1992, pp. 114-119*

[22] Lieberherr, K. J., Silva-Lepe, I., Xiao, C., "Adaptive Object-Oriented Programming using Graph-Based Customization", *CACM, Vol. 37, No. 5, May 1994, pp. 94-101*

[23] Lieberherr, K. J., "Demeter", *http://www.ccs.neu.edu/research/demeter/index.html*

[24] Liu, L., Zicari, R., Huersch, W., Lieberherr, K. J., "The Role of Polymorphic Reuse Mechanisms in Schema Evolution in an Object-Oriented Database", *IEEE Transactions of Knowledge and Data Engineering, Vol. 9, No. 1, Jan.-Feb. 1997, pp. 50-67*

[25] Loomis, M. E. S., "Object Versioning", *JOOP, Jan. 1992, pp. 40-43*

[26] Matthijs, F., *et al.*, "Aspects should not Die", *Proceedings of the AOP Workshop at ECOOP '97, 1997*

[27] Mens, K., Lopes, C., Tekinerdogan, B., Kiczales, G., "Aspect-Oriented Programming Workshop Report", *ECOOP '97 Workshop Reader, LNCS 1357, pp. 483-496*

[28] Mezini, M., Lieberherr, K. J., "Adaptive Plug-and-Play Components for Evolutionary Software Development", *Proceedings of OOPSLA 1998, ACM SIGPLAN Notices, Vol. 33, No. 10, Oct. 1998, pp.97-116*

[29] Monk, S. & Sommerville, I., "Schema Evolution in OODBs Using Class Versioning", *SIGMOD Record, Vol. 22, No. 3, Sept. 1993, pp. 16-22*

[30] Pazzi, L., "Explicit Aspect Composition by Part-Whole Statecharts", *Proceedings of the AOP Workshop at ECOOP '99, 1999*

[31] Pulvermueller, E., Klaeren, H., Speck, A., "Aspects in Distributed Environments", *Proceedings of GCSE 1999, Erfurt, Germany (to be published by Springer-Verlag)*

[32] Rashid, A., Sawyer, P., "Facilitating Virtual Representation of CAD Data through a Learning Based Approach to Conceptual Database Evolution Employing Direct Instance Sharing", *Proceedings of DEXA '98, LNCS 1460, pp. 384-393*

[33] Rashid, A., "SADES - a Semi-Autonomous Database Evolution System", *Proceedings of PhDOOS '98, ECOOP '98 Workshop Reader, LNCS 1543*

[34] Rashid, A., Sawyer, P., "Toward 'Database Evolution': a Taxonomy for Object Oriented Databases", *In review at IEEE Transactions on Knowledge and Data Engineering*

[35] Rashid, A., Sawyer, P., "Evaluation for Evolution: How Well Commercial Systems Do", *Proceedings of the First OODB Workshop, ECOOP '99, pp. 13-24*

[36] Rashid, A., Sawyer, P., "Dynamic Relationships in Object Oriented Databases: A Uniform Approach", *Proceedings of DEXA '99, LNCS 1677, pp. 26-35*

[37] Rashid, A., Sawyer, P., "Transparent Dynamic Database Evolution from Java", *Proceedings of OOPSLA 1999 Workshop on Java and Databases: Persistence Options*

[38] Rashid, A., Sawyer, P., Pulvermueller, E., "A Flexible Approach for Instance Adaptation during Class Versioning", *To Appear in Proceedings of ECOOP 2000 OODB Symposium*

[39] "SADES Java API Documentation", *http://www.comp.lancs.ac.uk/computing/users/marash/research/sades/index.html*

[40] Skarra, A. H. & Zdonik, S. B., "The Management of Changing Types in an Object-Oriented Database", *Proceedings of the 1st OOPSLA Conference, Sept. 1986, pp. 483-495*

[41] Suzuki, J., Yamamoto, Y., "Extending UML for Modelling Reflective Software Components", *Proceedings of UML '99*

Contextualization of OODB Schemas in CROME

Olivier Caron, Bernard Carré, and Laurent Debrauwer

{Olivier.Caron, Bernard.Carre, Laurent.Debrauwer}@lifl.fr
Laboratoire d'informatique Fondamentale de Lille, UPRESA CNRS 8022
Université des Sciences et Technologies de Lille
59655 Villeneuve d'Ascq cedex, France

Abstract. View mechanisms, widely used in the relational databases, pose new questions in the object model which captures much more semantics. In this paper, we will focus on inheritance and inter-objects relationships, two main semantic aspects of the object model. Like in the relational model, most of the current works about object-oriented views assume a fine granularity of the views. View classes are defined by the application of a query operator to one or two classes of the base schema. View schemas are defined as sets of view classes. These sets are explicitly chosen by the database administrator.
We present the solutions produced by the application of our CROME model. In CROME, view classes extend the descriptions of the domain objects supplied by the base schema. The relationships introduced in the base schema are shared and preserved in view schemas. By adapting them locally, each view schema contextualize these relationships. We will show that this contextualization of the base schema gives it generic properties which enforce a stronger coherence of the view schemas.

1 Introduction

The external schema level introduced in the ANSI/X3/SPARC[1] architecture is now integrated in all relational database systems. It determines contextual information relevant to groups of users of the information system. View mechanisms are usually used to implement such external schemas. The base or conceptual schema is, in the relational model, composed of a set of relations. Views are also relations computed by operators of the relational algebra.

Object-oriented databases (OODBs) view mechanisms mainly adhere to the same principles as relational views ([2], [3], [4], [5]). In the object model, views acquire the status of class. Their extension is computed from a query and they are updatable in some systems [4]. They provide a specialized interface to base classes. However, the semantics of the object model is richer than the relational one and this pose new questions. One of the most complex questions concerns the insertion of a view class in the generalization/specialization hierarchy. This insertion is generally realized by a classifier [6]. Some works about object-oriented views have introduced the notion of external schemas also called view schemas ([3],[5]). A view schema is a set of classes and view classes explicitly chosen by

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 135–149, 2000.

the administrator of the database. The classifier computes the class hierarchy which structures these view schemas.

However, these view mechanisms remain based on a fine granularity of the views which apply to individual classes. We focus here on the contextualization of the base schema considered as a whole with its structuring. This requires to take into account not only individual classes but also the class hierarchy itself and the inter-objects relationships at both intensional and extensional levels.

We present the solutions produced by the application of our CROME model ([7], [8], [9]) to build OODB view schemas. In CROME, the description of the objects is extended within the view schemas ([5],[10]). The base schema introduces the domain objects, give them their common description while the view schemas extend this description of the objects for their clients needs. The relationships introduced in the base schema are shared by the view schemas which contextualize them locally ([11], [12]). This applies to both inheritance and inter-objects relationships. At the extensional level we introduce the notion of participation constraints which determine the necessary conditions for an instance to be relevant in a context. They take into account the referential integrity and the cardinality constraints of the relationships to compute extensions of view classes. This contextualization gives to the base schema generic properties relatively to the view schemas.

In this paper, we focus on Enterprise Information Systems (EIS). Within an enterprise, the clients associated with the different view schemas of the EIS match generally the functions of the enterprise. Each function supplies and handles the information which is specific to it within the EIS, relatively to the tasks it must realize. This specific information is described by the view schema which corresponds to the function. The base schema describes the information common to the whole set of functions.

In this paper, the different notions of the CROME model are illustrated with the example of the EIS of a products distribution company. The management function and the sales function are structuring the organization of the company. The domain objects are partners, suppliers, customers, and products which are either professional products or consumer products.

The paper is organized as follows. Section 2 introduces the base schema while section 3 presents view schemas. In section 4, we examine how the inheritance relationships are contextualized in the view schemas while in section 5, we study the contextualization of inter-objects relationships and related constraints. Section 6 compares our approach with related works and section 7 concludes.

2 Base schema

Within the base schema, the domain objects are introduced by a set of classes structured by the inheritance and inter-objects relationships. These classes are called domain classes and the intension of these classes owns the description of objects which is common and relevant to the whole set of functions. These

common descriptions are shared by the different functions through the domain objects.

Example: The base schema of the example is shown in Figure 1 using the graphical notation of UML [13]. The domain objects are introduced as well as their common description by a set of classes structured by the inheritance hierarchy. An inter-objects relationship between "Customer" and "Product" is also introduced.

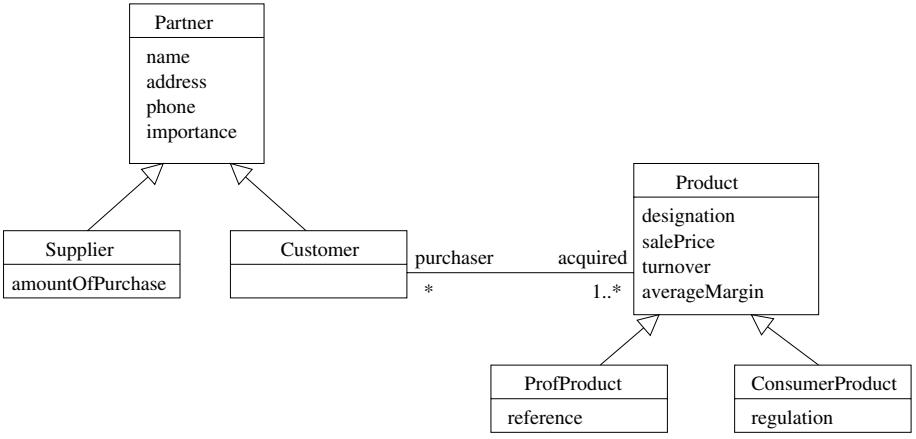


Fig. 1. Base schema

Formalization: Preliminary definitions
We define C as the set of classes of a schema.

$\forall$ C $\in C$, intension(C) denotes the set of the attributes of the class C.
$\forall$ C $\in C$, extension(C) denotes the set of the instances of the class C (including those of its subclasses).
The "subclass of" relationship, denoted by the symbol $<$, is defined as follows:
$\forall$ (C, C$'$) $\in C^2$, C $<$ C$' \Leftrightarrow$ intension(C) $\supseteq$ intension(C$'$) and
extension(C) $\subseteq$ extension(C$'$)

Base-schema
The base schema is composed of the set of domain classes ordered by the inheritance relationship. The set of the classes of the base schema is denoted by C_{base}. We call O the set of all instances of the domain classes, that is the union of the extension of every class of C_{base}.

3 View schemas

We introduce now, for each function of the enterprise, a view schema. The goal of such a view schema is to describe the information relevant to its corresponding

function, i.e. the information specific to this function and the common information whose description is inherited from the base schema.

3.1 Increments of description

From an intensional point of view, each domain object is enriched within each view schema. Its intension is extended with an increment of description relative to the specific information handled by the function corresponding to the view schema. The intension of an object is there defined as the union of its description in the base schema and this functional increment of description. Within a view schema, it is important to note that this functional increment is unique for every domain object[1].

Example: Figure 2 shows the functional increments corresponding to the sales function. We use a new UML stereotype increment in order to distinguish increments from classes. On this figure, we can see that the intension of "Customer" is enriched with the "turnover" attribute in the sales function. The "prospect/interest" relationship also enriches "Customer" and "Product" objects.

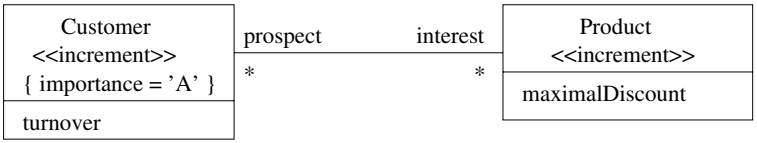


Fig. 2. Increments of description of the view schema corresponding to the sales function

Increments of description also specify the participation constraints of instances related to the view schema. For each function, we would like to determine the set of instances of the domain classes which own the information relevant to this function, and consequently participate to the corresponding view. In order to get this set for each domain class, we propose to use a participation constraint which will select its instances. This participation constraint takes the form of a predicate, called *participation predicate*, which applies to the state of the instance. When this predicate is verified, the instance participates to the view, i.e. is described by the corresponding increment of description and owns the related specific information. Because the predicate must be checked before the instance participates to the view, it only applies to the partial state (attributes and/or relationships) corresponding to the base schema. For each class of the base schema and for each view schema, there is only one participation predicate to select the instances of this class in this view schema[2].

[1] By default, this increment is empty

[2] By default, this predicate is a tautology.

Example: Figure 2 shows the participation constraint associated with the "Customer" object in the sales function[3]. It is a predicate based on the "importance" attribute. This attribute is introduced in the base schema as shown by Figure 1. This participation constraint means that only customers with an importance of type A are relevant to the sales function.

3.2 Status of view schemas

Each view schema extends the base schema by increasing the description of the domain objects and introducing participation constraints. So, each domain object which participates to the function corresponding to the view schema is there described by *a view class*. Its intension is the union of the intension of the corresponding class in the base schema, so called its *base class*, and the increment of description. Its extension is the set of the instances of the class which verify the participation predicate. In a view schema, for each class of the base schema, there is only one corresponding view class.

Example: Figure 3 shows the "Customer" and "Product" view classes in the view schema corresponding to the sales function. These classes match the description of a customer and a product under the point of view of the sales function. The other view classes have not been shown.

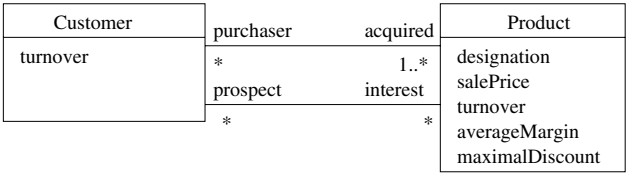


Fig. 3. View classes "Customer" and "Product" in the sales function

Therefore, view schemas are built on sets of view classes. We have now to study how these sets can be organized in respect to the inheritance and inter-objects relationships, as does the base schema.

4 Contextualization of the inheritance relationships of the base schema

In CROME, the base schema is shared as a whole graph of classes. The relationships between classes introduced in the base schema are shared in the different view schemas which contextualize them by adapting them locally. Inheritance and inter-objects relationships are both contextualized.

[3] We use the standard UML notation for constraints to denote the participation constraint.

4.1 Local inheritance of the intension of view classes

In a view schema, the increment of description of a class is inherited into the increment of description of its subclasses. Consequently, the base class hierarchy applies locally to view classes. From an intensional point of view, the inheritance relationships between classes of the base schema are contextualized in the view schemas.

Example: the increments of description of "Supplier" and "Customer" inherit from the increment of description of "Partner" in the management function as shown on Figure 4. This means that the view classes "Supplier" and "Customer" in the corresponding view schema inherit the "representative" attribute.

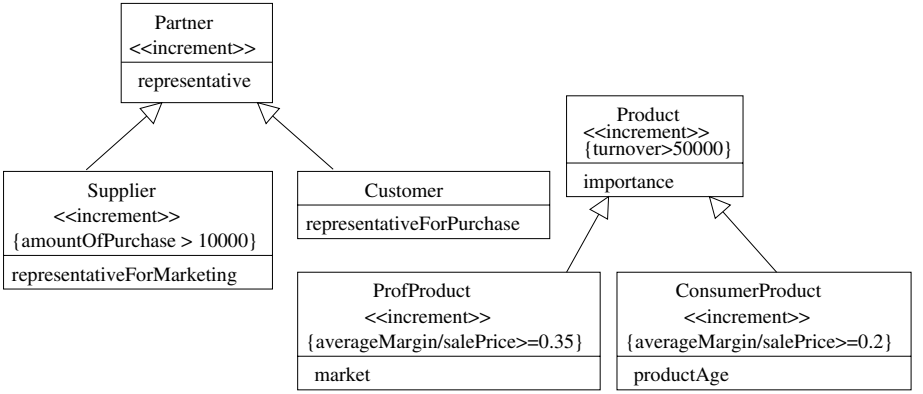


Fig. 4. Increments of description corresponding to the management function

Formalization

Set of functions

We use the functions to distinguish each view schema among the others. F is the set of symbols denoting the different functions.

Intension of view classes

$VC_f(C)$ denotes the view class corresponding to the class C in the function f, where $C \in C_{base}$ and $f \in F$.

$I_f(C)$ denotes the increment of description corresponding to the class C in the function f, where $C \in C_{base}$ and $f \in F$.

intension($VC_f(C)$), where $C \in C_{base}$ and $f \in F$, is defined as follows:

intension($VC_f(C)$) = intension(C) $\cup$ $I_f(C)$

Property 1: Let f be a function,

$\forall$ (C, C') $\in C^2_{base}$, C < C' $\Rightarrow$ intension($VC_f(C)$) $\supseteq$ intension($VC_f(C')$)

Indeed, we have the following property:
$\forall$ (C, C') $\in C^2_{base}$, C < C' $\Rightarrow$ intension(C) $\supseteq$ intension(C')
Due to the inheritance of increments of description, we have also the following property:
$\forall$ (C, C') $\in C^2_{base}$, C < C' $\Rightarrow$ I_f(C) $\supseteq$ I_f(C')
Consequently, due to the definition of the intension of a view class, this proves the property.

4.2 Relationships between extensions of view classes

In order to preserve the base class hierarchy structure within view schemas, the inheritance of the increments of description is not sufficient. We also need that the inclusion relationships between the extensions are respected within view schemas. In the base schema, an instance of a class belongs to all its superclasses. Similarly within a view schema, if an instance belongs to the extension of a view class, it must belong to the extension of all the related view superclasses.

The solution consists in giving to each instance a unique participation constraint to each view schema. This unique participation constraint is described as follows: to participate to a view, an instance must verify the participation predicate of the view class corresponding to its class. We have seen that, in a view schema, for each class of the base schema, there is only one corresponding view class. Since each instance has only one class in the base schema, its participation constraint is unique for each view schema. Thus, an instance that participates to a view belongs to the extension of the view class corresponding to its class and it belongs to the extension of the view classes corresponding to the superclasses of its class, in the manner of what occurs in the base schema.

This guarantees that the inclusion relationships between extensions of view classes are preserved in view schemas. Consequently, the structure of the base class hierarchy is preserved in the view schemas.

Finally, this solution guarantees that, if an instance participates to a view, it belongs to the extension of the view class corresponding to its class. Thus, the relationship between an instance and its class is preserved in views where this instance participates.

Example: in order to participate to the management function, a supplier must have an amount of purchase greater than 10000 euros (see Figure 4). This constraint is applied to the instances of "Supplier" to determine their participation to the management function, i.e. to determine if they belong to the extensions of the "Supplier" and "Partner" extensions. Figure 5 illustrates this example by showing the extension of these two classes.

Formalization
Functional state of an element of O
$\forall$ o $\in O$, class(o) denotes the class of o. This class belongs to C_{base}. $Pred_f$(C) denotes the participation predicate of VC_f(C), where C $\in C_{base}$ and f $\in F$.

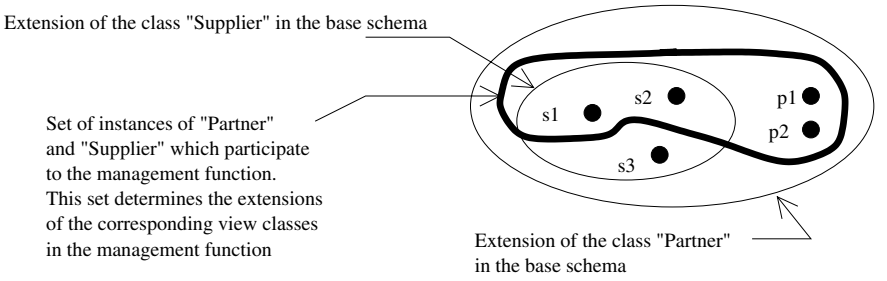


Fig. 5. Extensions of the classes "Partner" and "Supplier" and of their corresponding view classes in the management function

The functional state of an object is the set of functions in which o participates. it is denoted by EF(o) where o $\in O$.
$\forall$ o $\in O$, EF(o) = { f $\in F$ such as Pred_f(class(o)) is verified for o}
Extension of view classes
extension(VC_f(C)), where C $\in C_{base}$, is defined as follows:
extension(VC_f(C)) = { o $\in$ extension(C) such as f $\in EF$(o)}

Property 2: let f be a function,
$\forall$ (C,C') $\in C^2_{base}$, C $<$ C' $\Rightarrow$ extension(VC_f(C)) $\subseteq$ extension(VC_f(C'))

Indeed, let o be an element of extension(VC_f(C)), we know that f $\in EF$(o) and o $\in$ extension(C)
Since C is a subclass of C', o $\in$ extension(C') and we can prove that o $\in$ extension(VC_f(C'))

Property 3: let f be a function,
$\forall$ (C,C') $\in C^2_{base}$, C $<$ C' $\Rightarrow$ VC_f(C) $<$ VC_f(C')

This property is a direct consequence of properties 1 and 2.

4.3 Inheritance of the participation constraints

We know that an instance belonging to a view class belongs also to the extension of the superclasses (which are view classes) of this view class. This is always true even if this instance does not verify the participation predicate of these superclasses, which is not coherent. The solution is to define the participation predicate of a view class as the result of the conjunction of its own predicate and the predicate of all its superclasses. This conjunction guarantees that every instance belonging to a view class verifies its participation predicate. This applies also to instances of subclasses of this view class. We say that the participation predicate of a view class is inherited into its subclasses.

In view schemas, the inheritance relationship applies also for participation predicates, which reinforces its contextualization.

Example: in order to participate to the management function, a product must have a turnover greater than 50000 euros (see Figure 4). Professional products must verify this participation constraint due to the inheritance of the predicate of "Product" but they must also verify that their relative average margin is greater than 35% (i.e. averageMargin/salePrice$\geq$0.35). Consumer products must verify that their relative average margin is greater than 20%, the participation predicate of "Product" is also inherited into the "ConsumerProduct" view class.

As a conclusion, a view schema applies the base class hierarchy structure to the corresponding view classes in an isomorphic way, respecting their proper intensional and extensional constraints. This is a part of the contextualization process of the base schema by view schemas, which gives generic properties to inheritance relationships. We will see now how the base inter-objects relationships can also be contextualized in view schemas.

5 Contextualization of the inter-objects relationships of the base schema

We focus now on the sharing by the view schemas of the inter-objects relationships introduced in the base schema. Within a view schema, the extremities of such an inter-objects relationship are the view classes whose base classes are the extremities of this relationship in the base schema. The extension of these view classes can be a subset of the extension of their base classes. This may pose a problem regarding the respect of referential integrity and cardinality constraints within the view schema. These constraints are associated with the semantics of the relationships and we want to share the relationships in the view schemas with their entire semantics. The differences between the extensions in the base schema and in one of the view schemas may result in a violation of the constraints in this view schema, whereas these constraints are still verified in the base schema. We will see that the solution to keep these constraints verified in view schemas will lead us to a contextualization of the inter-objects relationships.

5.1 Referential integrity constraints

Within a view schema sharing a relationship of the base schema, the extremities of some occurrences could not be present as a result of the differences of extensions between view classes and their base classes. This results in a violation of referential integrity constraints in this view schema.

The solution consists in using only occurrences whose extremities are present in this view schema. The sharing is then reduced by taking into account the extensions of the view classes located at the extremities of the relationships.

Example: the relationship of the base schema "purchaser/acquired" between "Customer" and "Product" is shared by the view schema corresponding

to the management function. Figure 6 shows this relationship within the base schema with an example of occurrences of this relationship between instances of the classes "Customer" and "Product".

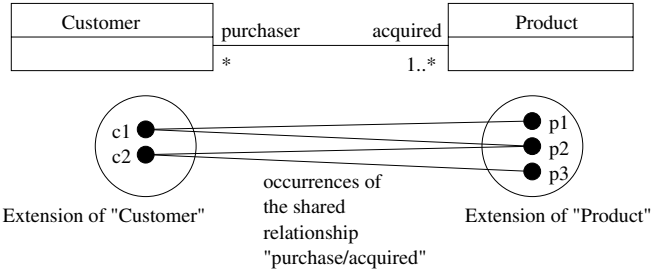


Fig. 6. Relationship "purchaser/acquired" introduced in the base schema

In the management context (see Figure 7), the view class "Product" introduces a participation constraint (see Figure 4) so the instance "p3" does not belong to its extension. Consequently, the occurrence between "c2" and "p3" is not selected within the view schema. This guarantees that the referential integrity constraints are verified in the involved context as shown on Figure 7.

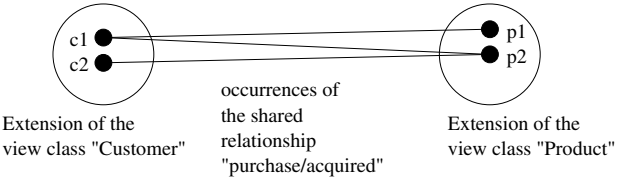


Fig. 7. Relationship "purchaser/acquired" and extensions of the view classes "Customer" and "Product" in the management function

Formalization

Respect of the referential integrity of shared relationships in view schemas

R_{base} denotes the set of inter-objects relationships introduced in the base schema. Let R(C,C') be a relationship between C and C', two classes of C_{base}. occ_{base}(R(C,C')) denotes the set of occurrences of R(C,C') in the base schema. The referential integrity in the base schema is formulated as follows:
$\forall$ R $\in$ R_{base}, occ_{base}(R(C,C')) $\subseteq$ extension(C)$\times$extension(C') where (C,C') $\in$

C^2_{base}.
$occ_f(R(C,C'))$ denotes the set of occurrences of R(C,C') in the view schema associated with the function f (f $\in F$). This set is obtained as follows:
$occ_f(R(C,C')) = occ_{base}(R(C,C')) \cap extension(VC_f(C)) \times extension(VC_f(C'))$
This set guarantees that the referential integrity is verified in this view schema.

5.2 Cardinality constraints

In the same way as referential integrity constraints, the cardinality constraints can be verified in the base schema without being verified within one of the view schemas. The reason comes from the differences between the extension of the view classes and the extension of their base classes. More precisely, this concerns the minimal cardinality which cannot be verified. This results from the fact that only a subset of the occurrences of the relationship is used in the view schema in order to enforce the referential integrity.

If, at one time, the cardinality constraints of a relationship are verified within the base schema and not within one of the view schemas, we do not want to trigger an exception. In fact, the goal of relationship sharing is not to increase the constraints associated with the relationships but to use them with their complete semantics. Consequently, the only way to verify the minimal cardinality constraint of a shared relationship in every view schema is to exclude from it all instances that cause the violation of this constraint. This adds a new modality to participation constraints.

Example: we continue with the example of the relationship called "purchaser/acquired" introduced in the base schema between "Customer" and "Product" (see Figure 6). This relationship introduces a minimal cardinality of one for the instances of "Customer". So, the instances of "Customer" that do not verify this minimal cardinality of one in the management function do not participate to the corresponding view. This can occur since the management function introduces participation constraints for the instances of "Product" (see Figure 4). Figure 8 shows this case for the instance "c2" of "Customer". There is no occurrence of "purchaser/acquired" that has "c2" as one of its extremities since the products "p2" and "p3" do not participate to the management function. Consequently, "c2" does not participate to this context. The fact that such a customer does not participate to the management function is justified by the semantics of a customer. This semantics contains the fact that a customer has purchased at least one product. A customer who has not purchased any product relevant to the management function is not considered as a customer relevant to this function.

The inter-objects relationships of the base schema own generic properties due to their contextualization in the view schemas. This contextualization is the result of their sharing which enforces the referential integrity and cardinality constraints in accordance with the extension of the view classes. Enforcing the referential integrity constraints leads to a restriction of the set of occurrences of a relationship. Enforcing the cardinality constraints creates a new participation

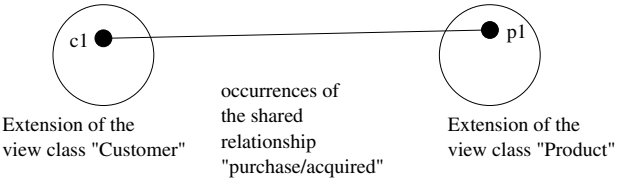


Fig. 8. Relationship "purchaser/acquired" and extensions of the view classes "Customer" and "Product" in the management function

constraint of the instances whose class is located at one of the extremities of the relationship.

Formalization

Respect of the minimal cardinality constraints of shared relationships in view schemas

We define, for the relationship R(C,C') $\in$ R_{base}((C,C') $\in$ C^2_{base}), the following cardinalities:

card-min$_C$(R(C,C')) is the minimal cardinality of R(C,C') for the extremity C.

card-min$_{C'}$(R(C,C')) is the minimal cardinality of R(C,C') for the extremity C'.

Minimal cardinality constraints are verified in the base schema if and only if:

$\forall$ R(C,C') $\in$ R_{base} where (C,C') $\in$ C^2_{base},

$\forall$ o $\in$ extension(C),

card-min$_C$(R(C,C')) $\leq$ $\|\{$(u,v) $\in$ occ$_{base}$(R(C,C')) such as u=o$\}\|$

$\forall$ o' $\in$ extension(C'),

card-min$_{C'}$(R(C,C')) $\leq$ $\|\{$(u,v) $\in$ occ$_{base}$(R(C,C')) such as v=o'$\}\|$

The same constraints must be checked in each view schema, that is for every view schema associated with the function f (f $\in$ F):

$\forall$ R(C,C') $\in$ R_{base} where (C,C') $\in$ C^2_{base},

$\forall$ o $\in$ extension(VC$_f$(C)), card-min$_C$(R(C,C')) $\leq$ $\|\{$(u,v)$\in$occ$_f$(R(C,C')) such as u=o$\}\|$

$\forall$ o' $\in$ extension(VC$_f$(C')), card-min$_{C'}$(R(C,C')) $\leq$ $\|\{$(u,v)$\in$occ$_f$(R(C,C')) such as v=o'$\}\|$

These constraints lead to a new definition of the functional state:

$\forall$o$\in$ O, EF(o)=$\{$ f $\in$ F such as:

- Pred$_f$(class(o)) is verified for o
- $\forall$ R(C,C') $\in$ R_{base} such as (C,C') $\in$ C^2_{base} and class(o) $<$ C,
  card-min$_C$(R(C,C')) $\leq$ $\|\{$ (u,v)$\in$occ$_f$(R(C,C')) such as u=o$\}\|$

$\}$

This definition has the drawback of using occ$_f$(R(C,C')) which is defined using the extension of view classes. We can easily rewrite this new definition of the

functional state:
$\forall o \in O, EF(o) = \{$ f $\in F$ such as:
- $\text{Pred}_f(\text{class}(o))$ is verified for o
- $\forall$ R(C,C') $\in R_{base}$ such as (C,C') $\in C^2_{base}$ and class(o) subclass of C,

card-$\min_C$(R(C,C')) $\leq \| \{$ (u,v) $\in \text{occ}_{\text{base}}$(R(C,C')) such as u=o and f $\in EF(v)\} \|$
$\}$

6 Comparison with related works

We have shown that CROME is a view approach which gives to the base schema generic properties relatively to the view schemas. These properties are the result of the sharing and the contextualization of the relationships of the base schema within the view schemas (inheritance and inter-objects relationships).

In the literature about view mechanisms ([2], [3], [4], [5]), view classes are defined by the application of a query operator to one or two classes of the base schema. This definition takes into account this or these base classes to compute the view class but does not take into account the hierarchy of the base schema. This results in a fine granularity of view classes. The view schemas are an assembly of classes and view classes explicitly and freely chosen by the database administrator. In each view schema, the inheritance relationships are computed by a classifier ([6], [14]) based on the subsumption relationship of the model. Generally, the hierarchy computed by this classifier and the hierarchy of the base schema will not be isomorphic at all[4]. We tried here to show how far it is possible to preserve the base class hierarchy structure in the view schemas. This gives generic properties to the base schema, as a regular and invariant structure transversely to the view schemas and makes their design operation easier.

The second point concerns the preservation of the link between an instance and its class. In the base schema, each instance is described by its class which inherits the description of its superclasses. In CROME, within every view schema, this link is preserved and exists between an instance and the view class issued from its class. On the one hand, an instance is described only by this view class as a result of the isomorphism of the inheritance relationship between the base schema and the view schemas. On the other hand, this instance belongs to the extension of view classes issued from its class and to the extension of their superclasses. Moreover, this instance does not belong to the extension of any other class. This results from the choice for each instance of a unique participation constraint within each view schema.

In view mechanisms, the role of the class of an instance is not preserved at all. In a view schema, an instance can belong and be described by several view classes such that there exists no inheritance relationship between them. Indeed, the extension of each view class is computed by using the entire extension of its

[4] Unless view classes has been defined for this purpose [8]

base class without taking into account the class of each instance which belongs to this extension.

Finally, the integrity constraints associated with the inter-objects relationships are not taken into account by view mechanisms. In a view, one of the extremities of an occurrence of a shared relationship can be an instance which does not participate to this view. This results from the fact that all occurrences of a relationship are shared without taking into account selections of instances. There exists only some ad-hoc techniques that use computed attributes in order to resolve the problems of referential integrity constraints[2].

7 Conclusion

We have proposed the CROME model in order to build view schemas which take into account the base schema at a level of granularity higher than in others works about object-oriented views. This is justified by the richer semantics of the object model and in particular, by the inheritance and inter-objects relationships. In CROME, we have seen that these two relationships are shared by the whole set of view schemas. So, these view schemas are isomorphic to the base schema. Moreover, these relationships are contextualized by taking into account the local aspects of the view schemas. All this give a true genericity to the base schema as a graph of classes.

CROME has been implemented as a design tool[15] of the base schema and view schemas[5]. It can be used with any OODB system having a binding with Java. This tool has been adapted for two OODB systems: PSE/PSE pro[6] and POET[7]. It contains a design browser which takes into account the whole set of schemas (base and view schemas) of the database. This browser checks the coherence of the generated schemas to avoid some design mistakes. The generation of the Java schemas relies on an architecture of interfaces packages which map the view schemas.

References

1. ANSI/X3/SPARC. Study Group on Database Management Systems. In *ACM Sigmod*, 1975.
2. A. Geppert, S. Scherrer, and K. Dittrich. Derived Types ans Subschemas: Towards Better support for Logical Data Independence in Object-Oriented Data Models. Technical report, Institut fur Informatik., Universitat Zurich, 1993.
3. E.A. Rundensteiner. A Methodology for Supporting Multiple Views in Object-Oriented Databases. In *Proceedings of the 18th International Conference on Very Large Databases (VLDB'92)*, pages 187–198, August 1992.
4. M.H. Scholl, C. Laasch, and M. Tresch. Updatable Views in Object-Oriented Databases. In *Proceedings of the Deductive and Object-Oriented Databases, Second International Conference*, pages 189–205, December 1991.

[5] This tool is available on `http://www.lifl.fr/DOWNLOAD/`

[6] `http://www.objectdesign.com`, PSE is a registered mark of Object Design, Inc

[7] `http://www.poet.com`, POET is a registered mark of POET software corporation

5. G. Guerrini, E. Bertino, B. Catania, and J. Garcia-Molina. *A Formal Model of Views for Object-Oriented Database Systems . Theory and Practice of Object Systems*, chapter 3. John Wiley, New York, 1997.
6. A. Borgida, R. Brachman, D. McGuinness, and L.A. Resnick. CLASSIC: a Structural Data Model for Objects. *SIGMOD Record*, 18(2):59–67, 1989.
7. L. Debrauwer, B. Carré, and G. Vanwormhoudt. Un cadre de conception par contextes fonctionnels de systèmes d'information à objets. In *XVème Congrès INFORSID*, Toulouse, juin 1997.
8. L. Debrauwer. *Des vues aux contextes pour la structuration fonctionnelle de bases de données à objets en CROME*. PhD thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, décembre 1998.
9. G. Vanwormhoudt, B. Carré, and L. Debrauwer. Programmation par objets et contextes fonctionnels. Application de CROME à Smalltalk. In *LMO'97*. R.Ducournau et S.Garlatti, octobre 1997.
10. E.A. Rundensteiner. A Transparent Object-Oriented Schema Change Using View Evolution. In *Proceedings of the IEEE international Conference on Data Engineering*, Taipei, March 1995.
11. L. Al-Jadir, G. Falquet, and M. Léonard. Context Versions in an Object-Oriented Model. In *Proceedings of the DEXA Conference*, 1993.
12. L. Al-Jadir, A. Le Grand, M. Léonard, and O. Parchet. Contribution to the Evolution of Information Systems. In T.W. Olle A.A. Verrijn-Stuart, editor, *Methods ans Associated Tools for the Information Systems Lifecycle,IFIP*. Elsevier, 1994.
13. G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.
14. E.A. Rundensteiner. A Classification Algorithm for Supporting Multiple Views in Object-Oriented Databases. In *Proceedings of the ACM third International Conference on Information and Knowledge Management (CIKM'94)*, pages 18–25, November 1994.
15. O. Caron, B. Carré, and L. Debrauwer. CromeJava: une implémentation du modèle crome de conception par contextes pour les bases de données à objets en Java. In *Proceedings of LMO'00*, January 2000.

Quality-Based Synchronization Methods of Multimedia Objects

Naokazu Nemoto, Katsuya Tanaka, and Makoto Takizawa

Dept. of Computers and Systems Engineering
Tokyo Denki University
Ishizaka, Hatoyama, Saitama 350-0394, Japan
E-mail {nemoto, katsu, taki}@takilab.k.dendai.ac.jp

Abstract. It is critical for applications to obtain enough quality of service (QoS) from multimedia objects. Not only states but also QoS of objects are changed by performing the methods on the objects. The objects are required to be consistent in presence of multiple transactions. We discuss new types of equivalent and conflicting relations among methods with respect to QoS. We also introduce two types of locking modes, serialization and mutual exclusion modes to synchronize concurrent accesses to objects based on the QoS relations.

1 Introduction

Distributed applications manipulate kinds of multimedia objects. The service supported by the object is characterized by parameters showing the *quality of service* (*QoS*) like frame rate. Not only the state but also QoS of the object are changed by performing the methods. Relations among the methods are so far discussed with respect to the states of the objects. For example, a pair of methods are *compatible* if states obtained by performing the methods in any order are the same [1]. Here, suppose that a state s_2 is obtained by dropping some frames in a state s_1 of a multimedia object. If s_2 satisfies QoS required by the applications, s_2 is considered to be equivalent with s_1 even if $s_1 \neq s_2$.

It takes a longer time to perform a method on a multimedia object since larger amount of structured data are manipulated. The throughput of the system is decreased if multiple methods are issued by multiple transaction and each of the methods are mutually exclusively performed. According to the synchronization theories [1], a pair of conflicting methods on an object are serially performed by locking the object. A pair of compatible methods can be performed in any order. In addition, some compatible methods can be concurrently performed, e.g. a pair of *read* methods can be concurrently performed. However, *increment* and *decrement* of a *counter* object cannot be concurrently performed. Thus, some methods are required to be mutually exclusively performed. We newly introduce two orthogonal types of lock modes, *serialization* and *mutually exclusive modes* based on QoS. The serialization locks [1] are used to serialize the computation of conflicting methods while the mutually exclusive locks are used to mutually exclusively perform methods. In this paper, we discuss how these lock modes are related with respect to QoS.

In section 2, we present a system model. In section 3, we discuss conflicting relations among methods based with respect to QoS. In section 4, we discuss how to lock objects to keep the objects consistent with respect to QoS.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 151–160, 2000.

2 System Model

2.1 Object-oriented model

A system is composed of multiple objects. An object is an encapsulation of data and methods for manipulating the data. Applications can obtain service only through the methods supported by the objects. There are two types of objects, *classes* and *instances*. A class c is composed of *attributes* $A_1, \ldots, A_m$ $(m \geq 0)$ and *methods* $op_1, \ldots, op_l$ $(l \geq 1)$. An instance o is created from the class c. The values of the instance o are changed only through the methods. A tuple $\langle v_1, \ldots, v_m \rangle$ of values is a *state* of the instance o where each v_i is a value taken by A_i Let *objects* show *instances* in this paper as used in Java [16] and C++ [14, 15]. Each object has one state at a time. A *state* of a class means a state of an object of the class.

A new class c_2 can be derived from an existing class c_1. Here, c_2 *inherits* the attributes and the methods from c_1. The new class c_2 can support additional attributes and methods. c_2 is a *subclass* or *derived* class of c_1. A class c can be composed of *component* classes $c_1, \ldots, c_n$. Let $c_i(s)$ denote a projection of a state s of the class c to c_i. For example, a class *Karaoke* is composed of three component classes, *music*, *words*, and *background*. The *background* class is furthermore composed of *car*, *tree*, and *cloud* classes.

On receipt of a request of a method *op*, *op* is performed on an object o. Let $op(s)$ and $[op(s)]$ denote a state and response obtained by performing *op* on a state s of o, respectively. Here, $op_1 \circ op_2$ and $op_1 \parallel op_2$ show that a pair of methods op_1 and op_2 are serially and concurrently performed, respectively.

2.2 QoS model

Applications obtain service from an object only through the methods of the object. Each service is characterized by parameters like level of resolution. These parameters are *quality of service* (QoS) supported by the object o.

The *scheme* of QoS is a tuple of attributes $\langle a_1, \ldots, a_m \rangle$ $(m \geq 1)$. Let $\mathrm{dom}(a_i)$ be a *domain* of an attribute a_i, i.e. a set of possible values to be taken by a_i $(i = 1, \ldots, m)$. For example, $\mathrm{dom}(colour)$ is a set of colours. A QoS *instance* q of the scheme $\langle a_1, \ldots a_m \rangle$ is given in a tuple of values $\langle v_1, \ldots, v_m \rangle \in \mathrm{dom}(a_1) \times \ldots \times \mathrm{dom}(a_m)$. Let $a_i(q)$ show a value v_i of an attribute a_i in the QoS instance q. Let S be a set of possible QoS instances. A QoS value v_1 *precedes* another one v_2 $(v_1 \succeq v_2)$ in $\mathrm{dom}(a_i)$ if v_1 shows better QoS than v_2. For example, $120 \times 100 \preceq 160 \times 120$ [pixels] for the attribute *resolution*. Let A be a subset $\langle b_1, \ldots, b_k \rangle$ of the QoS scheme $\langle a_1, \ldots, a_m \rangle$ where $b_j \in \{a_1, \ldots, a_m\}$ $(j = 1, \ldots, k)$ and $k \leq m$. A QoS *instance* q_1 of a scheme A_1 *partially dominates* q_2 of A_2 iff $a(q_1) \succeq a(q_2)$ for every attribute a in $A_1 \cap A_2$. For example, $\langle 160 \times 120$ [pixels], 1024 [colors], 15 [fps]$\rangle$ partially dominates $\langle 120 \times 100$ [pixels], 512 [colors]$\rangle$. q_1 *dominates* q_2 $(q_1 \succeq q_2)$ iff q_1 partially dominates q_2 and $A_1 \supseteq A_2$. Let S be a set of QoS *instances* whose schemes are not necessarily the same. A QoS instance q_1 is *minimal* in S iff there is no QoS instance q_2 in S such that $q_2 \preceq q_1$. q_1 is *minimum* iff $q_1 \preceq q_2$ for every q_2 in S. q_1 is *maximal* iff there is no q_2 in S such

that $q_1 \preceq q_2$. q_1 is *maximum* iff $q_2 \preceq q_1$ for every q_2 in S. A *least upper bound* (*lub*) $q_1 \cup q_2$ is some QoS instance q_3 in S such that 1) $q_1 \preceq q_3$ and $q_2 \preceq q_3$, and 2) there is no instance q_4 in S where $q_1 \preceq q_4 \preceq q_3$ and $q_2 \preceq q_4 \preceq q_3$.

An application requires an object to support some QoS which is referred to as *requirement* QoS (RoS). Let r be an RoS instance. Here, suppose an object o supports a QoS instance q. If $q \succeq r$, the applications can obtain enough service from the object o and q *satisfies* r. Otherwise, q is *less qualified* than r.

Each state s of an object o supports a QoS instance denoted by $Q(s)$.

3 QoS Based Conflicting Relations

3.1 Equivalent relations

A class *movie* is composed of two component classes; *advertisement* and *content*. An object m_1 created from *movie* is also composed of an *advertisement* object a_1 and a *content* object c_1. Another *movie* object m_2 is a same as m_1 except that the *advertisement* object is a_2. An application does not care the difference between m_1 and m_2 since the application is interested in only the content of *movie*. Here, m_2 is considered to be *equivalent* with m_1. A class like *content* in which applications are interested is *mandatory*[Figure 1]. On the other hand, a class like *advertisement* is *optional*. In addition, m_1 and m_2 support the same level of QoS, i.e. $Q(m_1) = Q(m_2)$. Suppose a class c is composed of classes $c_1, \ldots,$

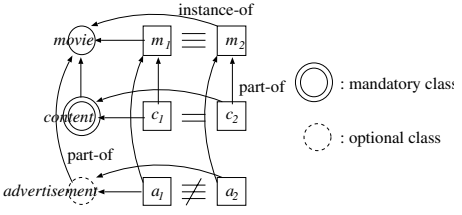


Fig. 1. Semantical equivalency.

c_m ($m \geq 0$). An application specifies whether each c_i is *mandatory* or *optional*. Every object o of c is required to include an object o_i of a mandatory class c_i. If c_i is optional, the object o may not include any object of c_i.

There are the following equivalent relations between a pair of states s_t and s_u of a class c:

- s_t is *state equivalent* with s_u iff $s_t = s_u$.
- s_t is *semantically equivalent* with s_u ($s_t \equiv s_u$) iff $s_t = s_u$ or $c_i(s_t) \equiv c_i(s_u)$ for every mandatory component class c_i of c.
- s_t is *QoS-equivalent* with s_u ($s_t \approx s_u$) iff s_t and s_u are obtained by degrading QoS of some state s of c, i.e. $Q(s_t) \cap Q(s_u) \preceq Q(s)$.
- s_t is *RoS-equivalent* with s_u on RoS R ($s_t -_R s_u$) iff $s_t \approx s_u$ and $Q(s_t) \cap Q(s_u) \succeq R$.
- s_t is *semantically RoS-equivalent* with s_u on RoS R ($s_t \equiv_R s_u$) iff $c_i(s_t) -_R c_i(s_u)$ or $c_i(s_t) \equiv_R c_i(s_u)$ for every mandatory class c_i of c.

[Example 1] Let K be an object created from the *Karaoke* class [Figure2]. The *Karaoke* class supports three methods *sound1*, *sound2*, and *sound3* which derive states s_1, s_2, and s_3, respectively, from a state s of K. Stereo sound of *music* m_1 is played while a *background* object *bg1* with a *words* object w_1 is displayed in the state s_1. *sound2* plays stereo sound of *music* m_2 while displaying the *words* object w_2 and a *background* object *bg2* which includes a *car* object. Suppose an application is interested only in *music* and *words*. Here, *music* and *words* are mandatory but *background* is optional in *Karaoke*. $s_1 \equiv s_2$ while $s_1 \neq s_2$ and $Q(s_1) = Q(s_2)$. Hence, *sound1* $\equiv$ *sound2*. On the other hand, *sound3* outputs a state s_3 of the object K, which shows only a monaural sound of *music* m_3 and *words* without *background*. Here, $s_1 \not\equiv s_3$ because $Q(s_1) \neq Q(s_3)$. $m_1 \approx m_3$. *sound1* supports a higher level of QoS than *sound3*. Suppose RoS R is that the application is not interested in the quality of sound. Here, $m_1 -_R m_2$. Furthermore, if *music* and *words* are mandatory, $s_1 \equiv_R s_3$ since $m_1 -_R m_3$. □

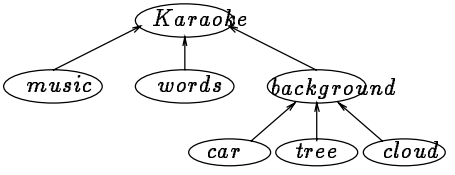


Fig. 2. Karaoke class.

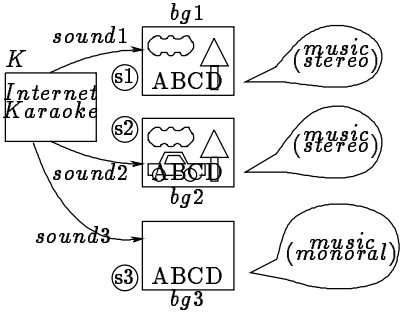


Fig. 3. Karaoke object.

There are the following equivalent relations between a pair of methods op_t and op_u of a class c :

- op_t is *state equivalent* with op_u iff $op_t(s) = op_u(s)$ for any state s of c.
- op_t is *semantically equivalent* with op_u ($op_t \equiv op_u$) iff $op_t(s) \equiv op_u(s)$ for every state s of c.
- op_t is *QoS-equivalent* with op_u ($op_t \approx op_u$) iff $op_t(s) \approx op_u(s)$ for every state s of c.
- op_t is *RoS-equivalent* with op_u on RoS R ($op_t -_R op_u$) iff $op_t(s) -_R op_u(s)$ for every state s of c.
- op_t is *semantically RoS-equivalent* with op_u on R ($op_t \equiv_R op_u$) iff $op_t(s) \equiv_R op_u(s)$ for every state s of c.

[Example 2] In the *Karaoke* object K 1 [Figure 3], the method *sound3* plays monaural sound obtained from *music* and displays *words* without *background*. *sound2* plays stereo *sound* while displaying *words* with the background object *bg2*. Here, s_2 and s_3 are states obtained by performing *sound2* and *sound3*, respectively, on a state s of K. Since, QoS of s_3 is different from s_2, i.e. $Q(s_2) \neq Q(s_3)$, *sound2* $\not\equiv$ *sound3*. Suppose there is such RoS R that the application

does not care whatever the background is and how qualified the music is. s_2 and s_3 satisfy R. That is, $sound2 \equiv_R sound3$. □

[**Proposition 1**] The following properties hold for every pair of methods op_t and op_u of a class c and RoS R:

- $op_t -_R op_u$ if $op_t \equiv_R op_u$.
- $op_t \approx op_u$ if $op_t -_R op_u$.
- $op_t \equiv_R op_u$ if $op_t \equiv op_u$. □

3.2 Compatible relations

In the traditional theories [1,8], a method op_t *conflicts* with another method op_u in an object o iff the result obtained by performing op_t and op_u depends on the computation order. Multimedia objects are characterized by QoS in addition to the states. There are the following compatible relations between a pair of methods op_t and op_u of a class c:

- op_t is *state-compatible* with op_u ($op_t \mid op_u$) iff $op_t \circ op_u(s) = op_u \circ op_t(s)$ for every state s of c.
- op_t is *QoS-compatible* with op_u ($op_t \parallel op_u$) iff $op_t \circ op_u(s) \approx op_u \circ op_t(s)$ for every state s of c.
- op_t is *semantically compatible* with op_u ($op_t \mid\mid\mid op_u$) iff $op_t \circ op_u(s) \equiv op_u \circ op_t(s)$ for every state s of c.
- op_t is *RoS-compatible* with op_u on RoS R ($op_t \mid_R op_u$) iff $op_t \circ op_u(s) -_R op_u \circ op_t(s)$ for every state s of c.
- op_t is *semantically RoS-compatible* with op_u on R ($op_t \mid\mid\mid_R op_u$) iff $op_t \circ op_u(s) \equiv_R op_u \circ op_t(s)$ for every state s of c.

Let "α-*compatibility* ($\diamond_\alpha$)" show some of the compatible relations, where $\alpha \in \{state, QoS, semantically, RoS, semantically\text{-}RoS\}$. For example, $\diamond_{QoS}$ shows $\parallel$ and $\diamond_R$ shows $\mid_R$. op_t α-*conflicts* with op_u ($op_t \not\diamond_\alpha op_u$) unless $op_t \diamond_\alpha op_u$. For example, op_t QoS-conflicts with op_u ($op_t \not\parallel op_u$) unless $op_t \parallel op_u$. $\diamond_\alpha$ is symmetric and is not transitive.

[**Example 3**] Suppose the *movie* object m is composed of a colored *background* object *bg* and a *car* object c, which supports methods *add*, *grayscale*, *mediascale*, and *reduce*. *add* adds a *car* object in the *movie*. *grayscale* degrades a colored video to a white-black gradation video. *mediascale* reduces a frame rate to half of the original one. *reduce* decreases a number of colors to 16 colors. Suppose an application performs *grayscale* on the object m after *add*, i.e. *add* ∘ *grayscale* and then an object m_1 is obtained. Here, m_1 is a white-black gradation video with the background and an object composed of the car. On the other hand, an object m_2 shows a colored *car* and the white-black background. Here, $Q(m_2) \neq Q(m_1)$. An object obtained by *add* ∘ *grayscale* supports a different level of QoS from *grayscale* ∘ *add*. *add* QoS-conflicts with *grayscale* (*add* $\not\parallel$ *grayscale*). By performing *grayscale* ∘ *add*, the response data of *add* shows white-black gradation *background* and colored *car*. However, the white-black gradation video is obtained by *add* ∘ *grayscale*. If the application is interested in a colored *car*, the response data obtained by *grayscale* ∘ *add* satisfies the application requirement

R. However, a response data obtained by *add* $\circ$ *grayscale* does not satisfy R. That is, *add* $\not|\!|_R$ *grayscale*.

Suppose the *movie* object m is displayed, whose QoS is $\langle$30 [fps], 256 [colors]$\rangle$ in Example 3. The application can obtain the same QoS by performing *mediascale* and *reduce* in any order. In any case, the application can get a state with 15 fps and 16 colors. *mediascale* || *reduce*. *reduce* ||| *mediascale* since *background* is optional. Suppose that RoS R shows "an application cares the color of the car". *reduce* $|_R$ *mediascale* but *add* $\not|_R$ *grayscale*. □

Suppose an application is not interested in how colorful movies are. A method *update* changes a colored *movie* to a monochromatic one. The application *displays* a colored *movie* object m. If *update* is performed on the state m of the *movie* object, the monochromatic version of m is seen. Since the application is not interested in the color, both of the versions are considered to satisfy RoS R. Hence, $Q([display(m)]) \cap Q([update \circ display(m)]) \succeq R$ and $Q(display \circ update(m)) = Q(update \circ display(m))$. *display* and *update* are RoS-compatible ($|_R$). However, *display* and *update* semantically conflict because $Q([update \circ display(m)]) \neq Q([display(m)])$. □

[**Proposition 2**] The following properties hold for every pair of methods op_t and op_u of a class c:

- op_t $|_R$ op_u if op_t $|||_R$ op_u.
- op_t || op_u if op_t $|_R$ op_u.
- op_t $|||_R$ op_u if op_t ||| op_u.

[**Proof**] It is straightforward from Proposition 1.□

4 Synchronization

4.1 Traditional locking protocol

According to the synchronization theories [1,8], every pair of conflicting methods issued by different transactions are required to be performed on the objects in the same order, i.e. *serializability*. In order to do that, an object o is locked before a method op_t is performed on o. If o is already locked for a method op_u conflicting with op_t, op_t blocks until the lock held by op_u is released. On the other hand, every pair of compatible methods can be performed on o in any order. Suppose a pair of transactions T_i and T_j issue methods op_t and op_u, to an object o_k, respectively. Here, T_i *precedes* T_j ($T_i \rightarrow T_j$) iff op_t and op_u conflict and op_t issued by T_i is performed on the object o_k before op_u issued by T_j. A collection of the transactions T_1, ..., T_m are *serializable* iff either $T_i \rightarrow T_j$ or $T_j \rightarrow T_i$ for every pair of transactions T_i and T_j, i.e. the transactions are totally ordered in the precedent relation "$\rightarrow$". It is well known that a collection of transactions are serializable if every transaction is two-phase locked [1].

Multiple conflicting methods cannot be concurrently performed on an object. It is still question whether or not multiple compatible methods can be concurrently performed. In some systems, some compatible methods like a pair of *read* methods can be concurrently performed. On the other hand, some compatible methods cannot be concurrently performed, i.e. the methods are to be mutually exclusively performed. For example, *add* and *subtract* are compatible on a *counter* object but cannot be concurrently performed. *Reduce* and *mediascale*

are RoS-compatible (*reduce* $|_R$ *mediascale*) but cannot be concurrently performed on the *movie* object as shown in Example 4. It is critical to definitely separate the lock concept to a pair of orthogonal concepts, locks for serialization and for mutual exclusion.

4.2 QoS based lock modes

If a pair of methods op_t and op_u α-conflict in an object o (op_t $\not\Diamond_\alpha$ op_u), the result obtained by performing op_t and op_u depends on the computation order of op_t and op_u. In the *movie* object m discussed in Example 3, the method *reduce* is *RoS*-compatible with *mediascale* on some RoS R (*reduce* $|_R$ *mediascale*). This means *reduce* and *mediascale* can be performed on the object m in any order for a given RoS R. However, *reduce* and *mediascale* cannot be concurrently reformed, i.e. mutually exclusive. On the other hand, a pair of *display* methods can be performed in any order since *display* is state-compatible with itself. In addition, multiple *display* methods can be concurrently performed because multiple transactions can view the *movie* object m at the same time. The traditional concurrency control theories [1] assume every pair of conflicting methods are mutually exclusive while compatible methods can be concurrently performed. However, some pair of compatible methods cannot necessarily be concurrently performed on a multimedia object. For example, unless some system allows multiple transactions to simultaneously view the *movie* object m, at most one *display* can be performed on m at a time. We define an *α-mutually exclusive* relation of methods as follows.

[**Definition**] A method op_t is *α-mutually exclusive* with op_u of a class c iff neither $op_t \parallel op_u$ is α-compatible with $op_t \circ op_u$ ($op_t \parallel op_u \Diamond_\alpha op_t \circ op_u$) nor $op_t \parallel op_u \Diamond_\alpha op_t \circ op_u$. □

Hence, we introduce two new orthogonal types of lock modes for a method op_t of an object o with respect to the α-compatibility and α-mutually exclusive relations :

1. *α-serialization* lock mode $\sigma_\alpha(op_t)$, and
2. *α-mutually exclusive* lock mode $\mu_\alpha(op_t)$.

The serialization locks are used to serialize conflicting methods issued by different transactions. The mutually exclusive locks are used to make methods mutually exclusively performed on an object. We define a conflicting relation among *α-serialization* and *α-mutually exclusive* lock modes.

[**Definition**] For every pair of methods op_t and op_u supported by an object o,

1. $\sigma_\alpha(op_t)$ *conflicts with* $\sigma_\alpha(op_u)$ and $\mu_\alpha(op_t)$ *conflicts with* $\sigma_\alpha(op_u)$ iff op_t α-conflicts with op_u.
2. $\mu_\alpha(op_t)$ *conflicts with* $\mu_\alpha(op_u)$ iff op_t is mutually exclusive with op_u. □

Let τ_1 and τ_2 be lock modes. τ_1 is compatible with τ_2 ($\tau_1 \Diamond \tau_2$) iff τ_1 does not conflict with τ_2. For example, the mutually exclusive mode $\mu_\alpha(display)$ is compatible with $\mu_\alpha(display)$ while the serialization mode $\sigma_\alpha(display)$ is compatible with $\sigma_\alpha(display)$. $\sigma_R(reduce)$ is compatible with $\sigma_R(mediascale)$ on RoS R since *reduce* $|||_R$ *mediascale*. However, $\mu_R(reduce)$ is not compatible with

$\mu_R(mediascale)$ on R since *reduce* and *mediascale* cannot be concurrently performed. Furthermore, $\mu_R(reduce)$ is not compatible with $\mu_R(mediascale)$. Every type of conflicting relation is assumed to be symmetric but not transitive.

Suppose that an object o is locked for a method op_t and another method op_u is issued to the object o. If $\sigma_\alpha(op_u)$ conflicts with $\sigma_\alpha(op_t)$, op_t blocks until op_u terminates. Suppose an object x supports a pair of methods op_1 and op_2 and another object y supports op_3 and op_4. A transaction T_1 issues op_1 to x and op_3 to y. Another transaction T_2 issues op_2 to x and op_4 to y. First, suppose op_1 is α-compatible with op_2 ($op_1 \diamond_\alpha op_2$) and $op_3 \diamond_\alpha op_4$. Here, $\sigma_\alpha(op_1)$ is compatible with $\sigma_\alpha(op_2)$ ($\sigma_\alpha(op_1) \diamond \sigma_\alpha(op_2)$) and $\sigma_\alpha(op_3) \diamond \sigma_\alpha(op_4)$. op_1 and op_2 can be performed on the object x and op_3 and op_4 on y in any order. For example, op_1 is performed after op_2 on x and op_4 is performed after op_3 on y.

Next, suppose $\mu_\alpha(op_1)$ conflicts with $\mu_\alpha(op_2)$ but $\mu_\alpha(op_3)$ is compatible with $\mu_\alpha(op_4)$. op_2 can be started after op_1 completes. op_1 and op_2 cannot be concurrently performed. However, op_3 and op_4 can be concurrently performed on the object y because $\mu_\alpha(op_3)$ is compatible with $\mu_\alpha(op_4)$.

[**Theorem 1**] A mutually exclusive mode $\mu_\alpha(op_t)$ conflicts with $\mu_\alpha(op_u)$ if a serialization mode $\sigma_\alpha(op_t)$ conflicts with $\sigma_\alpha(op_u)$.

[**Proof**] From the definitions, it is straightforward. □

If a method op_t α-conflicts with another method op_u in an object o, op_t and op_u cannot be concurrently performed. For example, a pair of the methods *add* and *mediascale* QoS-conflict and cannot be performed on the *movie* object m at the same time. That is, $\mu_{QoS}(add)$ conflicts with $\mu_{QoS}(mediascale)$.

If a transaction T issues a method op_t in the α-conflicting relation to an object o, o is locked according to the following protocol.

[**Locking protocol**]

1. The transaction T issues a lock request $\sigma_\alpha(op_t)$ to the object o.
2. If o is not locked in any mode conflicting with the serialization mode $\sigma_\alpha(op_t)$, o is locked in $\sigma_\alpha(op_t)$ and then is tried to be locked in a mode $\mu_\alpha(op_t)$.
3. If o is not locked in any mode conflicting with $\mu_\alpha(op_t)$, o is locked in the mode $\mu_\alpha(op_t)$ and then op_t is ready to be performed.
4. Otherwise, op_t blocks. □

4.3 Relation among lock modes

Figure 4 shows how conflicting relations of methods of an object o are related. Here, *State*, QoS, RoS, *Sem*, and *Sem-RoS* indicate sets of possible state-based, $QoS-$, RoS-, semantically, and semantically RoS-conflicting relations, respectively. Let S be a family of the sets {*State*, QoS, RoS, *Sem*, *Sem-RoS*}. Let α_1 and α_2 be sets in the set S, i.e. $\alpha_1 \in S$ and $\alpha_2 \in S$. In Figure 4, "$\alpha_1 \subseteq \alpha_2$" means that op_t α_2-conflicts with op_u if op_t α_1-conflicts with op_u for every pair of methods op_t and op_u. For example, a method op_t QoS-conflicts with another method op_u if op_t *Sem-conflicts* with op_u. Hence, $Sem \subseteq QoS$.

Let α_1 and α_2 be two types of conflicting relations. α_1 *dominates* α_2 ($\alpha_1 \succ \alpha_2$) iff $\alpha_1 \subseteq \alpha_2$ in Figure 4. For example, $Sem \succ QoS$ and $QoS \succ State$ from Figure 4. α_1 and α_2 are *uncomparable* ($\alpha_1 \parallel \alpha_2$) if neither $\alpha_1 \succ \alpha_2$ nor $\alpha_2 \succ \alpha_1$.

In Figure 4, $Sem \parallel RoS$. Let $C_\alpha(\tau)$ be a set of lock modes which are compatible with a lock mode τ on an α-conflicting relation. The following property holds on the dominant relation "$\prec$" :

[**Property**] For every lock mode τ and every pair of conflicting relations α_1 and α_2, $C_{\alpha_1}(\tau) \supseteq C_{\alpha_2}(\tau)$ if $\alpha_1 \prec \alpha_2$. □

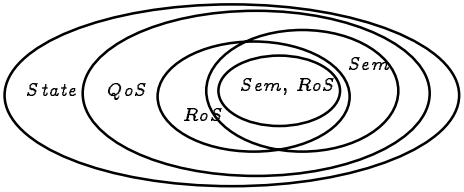


Fig. 4. Conflicting relations.

[**Definition**] A lock mode τ_1 on α_1 is *stronger* than another mode τ_2 on α_2 ($\tau_1 \succ \tau_2$) iff $C_{\alpha_1}(\tau_1) \subseteq C_{\alpha_2}(\tau_2)$. □

"$\tau_1 \succ \tau_2$" means that τ_1 conflicts with more number of lock modes than τ_2. The lock mode τ_1 is more restricted than τ_2. For example, *write* $\succ$ *read*.

Let α_1 and α_2 be types of RoS-conflicting relations on RoS R_1 and R_2, respectively. Here, it is that $R_1 \succ R_2$ (R_1 dominates R_2) means that R_1 shows a higher level of QoS than R_2. It is straightforward for the following theorem to hold from the definitions.

[**Theorem 2**] Let τ_1 and τ_2 be lock modes on RoS-conflicting relations α_1 and α_2, respectively, and R_1 and R_2 be RoS. $\tau_1 \succ \tau_2$ if $R_1 \succ R_2$. □

Suppose that R_1 and R_2 show monochromatic and colored movies, respectively, in Example 3. Let τ_1 and τ_2 be serialization lock modes of a method *grayscale* on RoS R_1 and R_2, respectively, i.e. $\tau_1 = \sigma_{R_1}(grayscale)$ and $\tau_2 = \sigma_{R_2}(grayscale)$. Here, $R_2 \succ R_1$. *grayscale* R_1-conflicts with *add* but is R_2-compatible with *add*. τ_1 is stronger than τ_2 ($\tau_1 \succ \tau_2$) since $C_{R_1}(\tau_1)$ (= {*mediascale*, *reduce*, *add*, *grayscale*}) $\supset C_{R_2}(\tau_2)$ (= {*mediascale*, *reduce*, *grayscale*}).

4.4 Implementation of lock modes

State-conflicting and *QoS*-conflicting relations among methods are defined in defining each object. In this paper, we assume there is one application manipulating distributed objects in the system. The application is composed of multiple transactions. A *Sem*-conflicting relation among methods is defined for each object, based on the semantics of the application. That is, mandatory and optional component classes are defined for a class. Each transaction requires its own *RoS*. It consumes plenty of computation power to compare arbitrary RoS instances. Hence, we assume some limited number of RoS instances are specified when the objects are defined in order to reduce the computation overhead. Each object maintains a table showing the conflicting relations among the lock modes. By using the conflicting table, it is decided if the method issued to the object can be performed on the object.

5 Concluding Remarks

We discussed novel types of relations among methods on the basis of QoS and the state of an object, i.e. state, QoS, RoS, and semantically RoS equivalent and conflicting relations in the object-based multimedia system. We presented the locking protocol to realize the *QoS*-conflicting relations, where new lock modes, serialization and mutually exclusive modes are introduced. By using the serialization and mutually exclusive locks, we can increase the performance of the system.

References

1. Bernstein, P. A., Hadzilacos, V., and Goodman, N., "Concurrency Control and Recovery in Database Systems," *Addison-Wesley*, 1987.
2. Cambell, A., Coulson, G., Garcfa, F., Hutchison, D., and Leopold, H., "Integrated Quality of Service for Multimedia Communication," *Proc. of IEEE InfoCom*, 1993, pp.732–793.
3. Campbell, A., Coulson, G., and Hutchison, D., "A Quality of Service Architecture," *ACM SIGCOMM Comp. Comm. Review*, Vol. 24, 1994, pp.6–27.
4. Gall, D., "MPEG: A Video Compression Standard for Multimedia Applications," *Comm. ACM*, Vol.34, No.4, 1991, pp.46–58.
5. Garcia-Molina, H. and Salem, K., "Sagas," *Proc. of ACM SIGMOD*, 1987, pp.249–259.
6. Kanezuka, T. and Takizawa, M., "QoS Oriented Flexibility in Distributed Objects," *Proc. of Int'l Symp. on Comm.* (*ISCOM'97*), 1997, pp.144–148.
7. Kanezuka, T. and Takizawa, M., "Quality-based Flexibility in Distributed Objects," *Proc. of 1st IEEE Int'l Symp. on Object-oriented Real-time Distributed Computing* (*ISORC'98*), 1998, pp.350–357.
8. Korth, H. F., Levy, E., and Silberschalz, A., "A Formal Approach to Recovery by Compensating Transactions," *Proc. of VLDB*, 1990, pp.95–106.
9. MPEG Requirements Group, "MPEG-4 Requirements," ISO/IEC JTC1/SC29/WG11 N2321, 1998.
10. Object Management Group Inc., "The Common Object Request Broker: Architecture and Specification, Rev2.0," 1995.
11. Owen, C. B. and Makedon, F., "Computed Synchronization For Multimedia Applications," *Kluwer Academic*, 1999.
12. Takizawa, M. and Yasuzawa, S., "Uncompensatable Deadlock in Distributed Object-Oriented Systems," *Proc. of IEEE ICPADS-92*, 1992, pp.150–157.
13. Yoshida, T. and Takizawa, M., "Model of Mobile Objects," *Proc. of DEXA'96* (*Lecture Notes in Computer Science*, *Springer-Verlag*, No. 1134), 1996, pp. 623–632.
14. Kruglinski, D. J., "Inside Visual C++," *Microsoft Preess*, 1997.
15. Stroustrup, B., "The C++ Programming Language (2nd ed.)," *Addison-Wesley*, 1991.
16. Grosling, J. and McGilton, H., "The Java Language Environment," *Sun Microsystems,* Inc., 1996.

Utilizing Fragmented Bandwidth in a Staggered Striping Multimedia System

Yang Pan and Wen-Chi Hou

Department of Computer Science, Southern Illinois University at Carbondale
Carbondale IL, 62901
hou@cs.siu.edu

Abstract. In this paper, we discuss the use of fragmented bandwidth to improve the performance of the staggered striping in a multimedia system. It is observed that potential disruptions can occur when non-consecutive idle disks are used for displaying multimedia objects. We have identified useful retrieval patterns and shown that with proper selections of fragmented disks and a simple buffering scheme disruptions can be easily eliminated.

1 Introduction

Recent advances in networking and multimedia technology have made it possible to provide concurrent services, such as on-line shopping, on-line auction, etc., to hundreds or even thousands of customers. Unlike conventional video rental stores, which can serve only a very limited number of viewers simultaneously, a multimedia-on-demand (MOD) system is intended to provide on-line, concurrent, convenient, and faster service to a large number of viewers. Besides videos, new multimedia applications, such as graphical modeling of nearby 3D objects in architectural buildings, urban city models, scientific visualization, etc., [19], and can also benefit from the technology of MOD systems.

In an MOD system, hundreds or even thousands of multimedia objects are stored on a storage server and are ready to be played out upon requests. Multimedia storage servers are generally connected to clients via high-speed networks so that continuous media such as video and audio can be played on users' stations. Viewers can also access multimedia objects simultaneously on the server. The goal of an MOD server is to serve as many viewers concurrently as possible.

Multimedia data can have varying and sometimes very high bandwidth requirements. For example, a 27 MBps (i.e., megabytes per second) bandwidth is required by NTSC for "network-quality" video [11], while a bandwidth of approximately 81 MBps is required for a HDTV-quality image [11]. Although the bandwidth of a modern magnetic disk drive can reach approximately 20 MBps [14], multimedia data are expected to have much higher bandwidths in the future [2].

Compression techniques [16], such as MPEG-1 and MPEG-2, have been applied successfully to multimedia objects like movie videos, to reduce the bandwidth requirements. However, these compression techniques generally are lossy and can result in loss of important information. Thus, they may not be acceptable to other

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 161–170, 2000.

objects like computer programs, scientific data, medical data, etc., which require high accuracy. Thus, one of the major challenges in building an MOD system is to support varying and perhaps high-bandwidth data with low-bandwidth disk drives.

Although individual disks may have large enough storage capability [13] and speed to display some compressed video objects, a bottleneck can easily arise when there are simultaneous requests for videos stored on the same disk. Since a bottleneck can result in a significant delay in display, it may not be desirable to store an entire video object on a single disk. Thus, how to place the data on disks to balance the workload and serve more requests without much delay has also emerged as an important issue.

The striping techniques [1,2,9,12] effectively overcome the bottleneck problem by declustering objects across multiple disks and using the aggregated bandwidth of disks to serve simultaneous requests. In addition, servers in the striping approach can store more video objects than in the replication methods. The staggered striping [2] is a variant of the striping techniques and is very promising because of its flexibility and capability in storing and displaying multimedia objects of variable bandwidths. However, as requests arrive and are served, the system could be left with dispersed variable-sized "holes" of idle disks, termed fragmented disks [2], during the display. This situation is similar to the memory fragmentation in the traditional operating systems. When there is no single hole that is large enough to meet the bandwidth requirement of the requested object, the request can not be served, even though the total number of idle disks in the system is enough. Berson et al. [2] pointed out potential bandwidth waste due to this bandwidth fragmentation. They have shown an example of how portions of subobjects, which are contiguous portions of an object, can be retrieved using fragmented disks and buffered until the entire subobjects are put together.

However, there were no discussions and analysis on how fragmented bandwidth can be utilized in general. In fact, we have found that the use of fragmented bandwidth is much more complex than the example they showed. Specifically, disruptions, termed as "hiccups" [2,4,9], can occur not only at the beginning of the display but also in the middle of display. Moreover, the buffer space needed to store incomplete pieces of subobjects can be very large if we do take advantage of some properties of the retrieval patterns. To the best of our knowledge, there have not been any in-depth studies on the utilization of fragmented bandwidth in the staggered striping. In this paper, we propose a method to solve the disruption problems when non-consecutive idle disks are used for display. We identified repeated patterns of retrieval that can be utilized to smooth the display along. With proper selection of idle disks and a simple buffering scheme, fragmented bandwidth can be fully utilized and disruptions can be eliminated

The rest of the paper is organized as follows. A review of related work is presented in Section2. In Section3, we demonstrate the disruption problems in the staggered striping and point out the existence of potential disruption patterns. In Section 4, a formal analysis of the disruption patterns is conducted and a solution to the disruption problems is presented. Section 5 is the conclusions.

2 Staggered Striping

The basic idea of the striping approach is to stripe the objects across clusters of disks so that the aggregate bandwidth of a cluster can match the bandwidth requirement of objects. An object, say X, is divided into a number of subjects Xi's that is intended to be the unit of transfer between the server and the viewer's station. Each subject is then declustered over a number M(X) of disks, where M(X) is called the degree of declustering, so that M(X) disks can be used simultaneously to satisfy the bandwidth requirement of the object. Each portion of the subject Xi declustered on a disk is called a fragment, denoted as Xi.j, $0 \leq j < M(X)$.

Stride k is defined to be the distance, in terms of the number of disk drives, between the first fragments of two consecutive subobjects Xi.0 and X(i+1).0. Notice that in the simple striping, k is M(X), while in the staggered striping, k can vary in value from 1 to D.

Objects, maybe of different bandwidths, are assigned to disks independently with the same stride. Fig. 1 illustrates a possible placement of objects X, Y, and Z with bandwidth requirements of 40, 80, and 60 MBps, respectively, in a system with 9 disks (with Bdisk = 20 MBps). X0 is stored on disks 0 and 1 (M(X) = 2), Y0 is on disks 2, 3, 4, and 5 (M(Z) = 4), and Z0 is on disks 0, 7, and 8 (M(Y) = 3). Note that the strides for all objects are 1.

disks	0	1	2	3	4	5	6	7	8
	$Z_{0.2}$		$Y_{0.0}$	$Y_{0.1}$	$Y_{0.2}$	$Y_{0.3}$		$Z_{0.0}$	$Z_{0.1}$
	$Z_{1.1}$	$Z_{1.2}$		$Y_{1.0}$	$Y_{1.1}$	$Y_{1.2}$	$Y_{1.3}$		$Z_{1.0}$
	$X_{0.0}$	$X_{0.1}$							
		$X_{1.0}$	$X_{1.1}$						

Fig. 1. Placement of objects on 9 disks

In order to display an object, say X, the system must first locate the M(X) adjacent disk drives that contain subobject X0 (i.e., disks 0 and 1). If these disk drives are idle, X0 can be retrieved and displayed immediately; otherwise, the display has to be delayed until these disks become idle. At the next time interval, it reads from the next M(X) disk drives (i.e., disks 1 and 2), by shifting k (i.e., 1 in this example) disks to the right.

As requests arrive and are served, the system is left with dispersed "holes" of idle disks, similar to the memory fragmentation problem in the traditional operating system. Since in the striping approach, all fragments of a subobject must be retrieved from disks at the same time. If the disks containing the requested subobject are not all idle at the same time, bandwidth fragmentation can occur. For example, consider the placement of objects in Fig. 1. again. Assume the system is to retrieve Y0 and Z0 at the moment and a request for X has just arrived. While there are two disks (1 and 6) idle, a request for object X can not be served until the display of Z completes, because disks 0 and 1 are not available at the same time. These free but non-consecutive idle disks, like 1 and 6, are called fragmented disks [2]. To utilize the bandwidth of fragmented disks, additional buffer space is used in [2].

While it seems simple to utilize fragmented bandwidth, the above example only illustrates a best scenario. In fact, we found that disruptions of display can occur at

any time when fragmented disks are used. To the best of our knowledge, such disruptions (that occur during the display) have not been addressed in the literature. We will describe this problem in detail in Section 3 and discuss the solution in Section 4.

3 Observations on the Disruptions

The quality of a display is severely compromised if disruptions occur during the display. Disruptions can occur when there are not enough idle disks to display objects. However, even with enough idle disks, there can still be disruptions if these idle disks are not consecutive, i.e., fragmented. In this section, we show how disruptions can occur.

In Fig. 2, we show how disruptions can occur in a staggered striping system with k = 1. Assume at t0, disks 0, 1, 2, and 4 are idle and are designated to serve a request for object X with M(X) = 4. Note that the four idle disks are not consecutive and the rest of disks are busy with other requests. For clarity, only fragments of object X retrieved at each time interval are indicated in the figure. Note that each disk stores a set of fragments coming from different subobjects. Fragments on a disk are read sequentially for the display of the object.

The rightmost column "completed subobjects" lists subobjects, all of whose fragments have been completely retrieved from disks at the corresponding time interval. As shown in the figure, a disruption occurred at t4, and X3 was not ready until t13. Note that other subobjects, such as X4, X5, X6, X7, X8, X9, X10, X11 and X12, are all ready before X3. A naive solution to eliminating disruptions is to delay the retrieval until there are enough consecutive idle disks in the system. Unfortunately, it can result in substantial delay as consecutive idle disks may not be available until one or more current requests are completed.

It is observed from Fig. 2 that although subobjects retrieved are out of sequence, there are still the same number of subobjects retrieved (i.e., 14 = D) in the first 14 time units as if we were using consecutive idle disks. In addition, the retrieval of fragments is not completely random, that is, no other fragments, except the fragments of the first 14 subobjects are read during the first 14 time units. Another important observation is that the set of disks used at t0 is used again at t14 to retrieve fragments of X14 and X15 (vs. X0 and X1 at t0). Consequently, the retrieval pattern appearing during the period t0 to t13 is repeated in the next and every successive 14 time units. This observation prompts the idea that if we can buffer a small amount of subobjects (e.g., the first 14 subobjects in the example) in advance, then the display may be able to be smoothed over without disruption. That is, while we are going to retrieve the next 14 subobjects in the next 14 time units, we display the previously buffered 14 subobjects.

Disk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	*completed subobjects*
t_0	$X_{0.0}$	$X_{0.1}$	$X_{0.2}$		$X_{1.3}$										
t_1		$X_{1.0}$	$X_{1.1}$	$X_{0.3}$		$X_{2.3}$									X_0
t_2			$X_{2.0}$	$X_{1.2}$	$X_{2.2}$		$X_{3.3}$								X_1
t_3				$X_{2.1}$	$X_{3.1}$	$X_{3.2}$		$X_{4.3}$							X_2
t_4					$X_{4.0}$	$X_{4.1}$	$X_{4.2}$		$X_{5.3}$						X_4
t_5						$X_{5.0}$	$X_{5.1}$	$X_{5.2}$		$X_{6.3}$					X_5
t_6							$X_{6.0}$	$X_{6.1}$	$X_{6.2}$		$X_{7.3}$				X_6
t_7								$X_{7.0}$	$X_{7.1}$	$X_{7.2}$		$X_{8.3}$			X_7
t_8									$X_{8.0}$	$X_{8.1}$	$X_{8.2}$		$X_{9.3}$		X_8
t_9										$X_{9.0}$	$X_{9.1}$	$X_{9.2}$		$X_{10.3}$	X_9
t_{10}	$X_{11.3}$										$X_{10.0}$	$X_{10.1}$	$X_{10.2}$		X_{10}
t_{11}		$X_{12.3}$										$X_{11.0}$	$X_{11.1}$	$X_{11.2}$	X_{11}
t_{12}	$X_{12.2}$		$X_{13.3}$										$X_{12.0}$	$X_{12.1}$	X_{12}
t_{13}	$X_{13.1}$	$X_{13.2}$		$X_{3.0}$										$X_{13.0}$	X_3 X_{13}
t_{14}	$X_{14.0}$	$X_{14.1}$	$X_{14.2}$		$X_{15.3}$										
				……											

Fig. 2. Retrieval of Object X Using Non-consecutive Idle Disks 0, 1, 2 and 4 with k = 1.

The values of k affect retrieval patterns when non-consecutive idle disks are used. Recalling that in Fig. 2, where k was set to 1, although a disruption occurred at t4, all the first 14 subobjects, X0, X1, …, X13, were completely retrieved by the end of t13, and a new cycle begins at t14. This repeated pattern can be identified easily and can be very useful in eliminating disruption.

Disk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	*completed subobjects*
t_0	$X_{0.0}$	$X_{0.1}$	$X_{0.2}$		$X_{1.2}$										
t_1			$X_{1.0}$	$X_{0.3}$	$X_{2.0}$		$X_{2.2}$								X_0
t_2					$X_{8.2}$	$X_{1.3}$	$X_{3.0}$		$X_{3.2}$						
t_3						$X_{9.2}$	$X_{2.3}$	$X_{4.0}$		$X_{4.2}$					
t_4								$X_{10.2}$	$X_{3.3}$	$X_{5.0}$		$X_{5.2}$			
t_5	$X_{6.2}$									$X_{11.2}$	$X_{4.3}$	$X_{6.0}$			
t_6	$X_{7.0}$		$X_{7.2}$									$X_{12.2}$	$X_{5.3}$		
t_7	$X_{13.2}$	$X_{6.3}$	$X_{8.0}$		$X_{9.0}$										
t_8			$X_{14.2}$	$X_{1.1}$	$X_{15.2}$		$X_{10.0}$								X_1
t_9					$X_{16.0}$	$X_{2.1}$	$X_{16.2}$		$X_{11.0}$						X_2
t_{10}							$X_{17.0}$	$X_{3.1}$	$X_{17.2}$		$X_{12.0}$				X_3
t_{11}									$X_{18.0}$	$X_{4.1}$	$X_{18.2}$		$X_{13.0}$		X_4
t_{12}	$X_{14.0}$										$X_{19.0}$	$X_{5.1}$	$X_{19.2}$		X_5
t_{13}	$X_{20.2}$		$X_{15.0}$										$X_{20.0}$	$X_{6.1}$	X_6
t_{14}	$X_{21.0}$	$X_{7.1}$	$X_{21.2}$		$X_{22.3}$										
		…													

Fig. 3. Retrieval of Object X Using Non-consecutive Idle Disks 0, 1, 2 and 4 when k = 2.

However, the situations may be a little more complex when k is set to other values. Let us consider Fig. 3, where k is set to 2. Although there are still four fragments of object X read in each time interval, some disks, such as disks 1, 3, 5, 7, 9, 11, and 13, have read only two fragments during the first 14 time units, while others read six fragments. Due to this imbalance in reading, the retrieval of those fragments on odd-numbered disks is considerably delayed. For example, fragments X7.1 and X7.3, on disks 1 and 3, respectively, were not retrieved until t14 and t15, while some other

higher-numbered fragments, like X15.2, X16.0, etc., were read much sooner than in the previous case (i.e., k = 1). Indeed, fragments on even-numbered disks are retrieved sooner, while much later for fragments on odd-numbered disks, prolonging the entire retrieval process by a factor of 2. This implies a very large buffer and long delay may be needed if we attempt to eliminate disruptions.

If we had used idle disks 0, 1, 2, and 5, assumed available, the situation would have been quite different. As shown in Fig. 4, with idle disks 0, 1, 2, and 5 designated at t0, all the first 7 subobjects are completely retrieved by the end of t6. A new cycle begins at t7 and repeats for every 7 time units. This pattern could be useful in our attempt to display objects continuously within a shorter time.

Disk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	*completed subobjects*
t_0	$X_{0.0}$	$X_{0.1}$	$X_{0.2}$			$X_{1.3}$									
t_1			$X_{1.0}$	$X_{0.3}$	$X_{1.2}$			$X_{2.3}$							X_0
t_2					$X_{2.0}$	$X_{2.1}$	$X_{2.2}$			$X_{3.3}$					X_2
t_3							$X_{3.0}$	$X_{3.1}$	$X_{3.2}$			$X_{4.3}$			X_3
t_4									$X_{4.0}$	$X_{4.1}$	$X_{4.2}$			$X_{5.3}$	X_4
t_5	$X_{6.3}$										$X_{5.0}$	$X_{5.1}$	$X_{5.2}$		X_5
t_6	$X_{6.2}$			$X_{1.1}$									$X_{6.0}$	$X_{6.1}$	X_1 X_6
t_7	$X_{7.0}$	$X_{7.1}$	$X_{7.2}$			$X_{8.3}$									
t_8			$X_{8.0}$	$X_{7.3}$	$X_{8.2}$			$X_{9.3}$							X_7
t_9					$X_{9.0}$	$X_{9.1}$	$X_{9.2}$			$X_{10.3}$					X_9
t_{10}							$X_{10.0}$	$X_{10.1}$	$X_{10.2}$			$X_{11.3}$			X_{10}
t_{11}									$X_{11.0}$	$X_{11.1}$	$X_{11.2}$			$X_{12.3}$	X_{11}
t_{12}		$X_{13.3}$									$X_{12.0}$	$X_{12.1}$	$X_{12.2}$		X_{12}
t_{13}	$X_{13.2}$			$X_{8.1}$									$X_{13.0}$	$X_{13.1}$	X_8 X_{13}
t_{14}	$X_{14.0}$	$X_{14.1}$	$X_{14.2}$			$X_{15.3}$									
t_{15}			$X_{15.0}$	$X_{14.3}$	$X_{15.2}$			$X_{16.3}$							X_{14}
			……												

Fig. 4. Retrieval of Object X Using Non-consecutive Idle Disks 0, 1, 2, and 5 when k = 2.

From Figures 3 and 4, we notice that with the proper choice of idle disks, if available, useful repeated patterns with a shorter period (less than D) may be found when k values are different than 1. In the next section, we will formally discuss how a pattern is affected by various factors, such as k, D, placement of subobjects, and idle disks, etc.

4 Pattern Analysis

The goal is to find the existence of a period of p time units, within which the next p successive subobjects can be completely retrieved using a given set of non-consecutive disks. For simplicity, we shall not mention the object of concern explicitly in the following discussion

Definition. The storage sequence for the fth fragments of subobjects m to n, denoted Sf(m, n), is a sequence of disk IDs that stores the fth fragments of successive subobjects from m to n.

Let j be the ID number ($0 \le j < D$) of the disk containing the fth fragment of the 0th subobject of X (i.e., X0.f). By definition of the stride k, disk $(j + k \times n) \bmod D$ must be the disk containing the fth fragment of the nth subobject. Thus, S0(0, n) consists of disks j , $(j + k \times 1) \bmod D$, …, $(j + k \times n) \bmod D$.

Definition. The retrieval sequence beginning with disk i for the period from tm to tn, denoted as Ri(tm, tn), is the sequence of disk IDs beginning with disk i that is used in successive time intervals from tm to tn to retrieve the object in the staggered striping.

Let i be the ID number ($0 \le i < D$) of one of the disks designated to serve a request for a certain object X at t0. Then at tn, we can infer that disk $(i + k \times n) \bmod D$ must be a disk serving X, because of the stride k. Consequently, Ri(t0, tn) is made of disks i, $(i + k \times 1) \bmod D$, …, $(i + k \times n) \bmod D$. Note that each disk designated to serve X at t0 has its own retrieval sequence.

Let ω be the largest common divisor of D and k. Then, both a storage and a retrieval sequence can be rewritten as i, $(i + k \times 1) \bmod D$, …, $(i + k \times D/\omega) \bmod D$, $(i + k \times (D/\omega + 1)) \bmod D$, …, etc. Since $(k \times D/\omega) \bmod D = 0$, $(i + k \times D/\omega) \bmod D$ becomes $(i + k \times 0) \bmod D$, $(i + k \times (D/\omega + 1)) \bmod D$ becomes $(i + k \times 1) \bmod D$, and so on. As a result, there can be only D/ω distinct disks in each storage or retrieval sequence of length greater than D/ω, and they are reused every D/ω time units.

Let im, $0 \le m < M(X)$, be a set of M(X) idle disks chosen to serve X at t0. As mentioned earlier, each disk stores a set of fragments from different subobjects (of an object) and those fragments on a disk are retrieved one at a time sequentially when the disk is used for the display of the object. It can be conceived that if the disks appearing in the sequences Rim(0, D/ω), $0 \le m < M(X)$, also appear the same number of times in sequences Sf(0, D/ω), $0 \le f < M(X)$, and vice-versa, then the first D/ω subobjects, nothing more and nothing less, are retrieved during the first D/ω period. Note that the sequences Sf(0, D/ω), $0 \le f < M(X)$, represent the first D/ω objects. Moreover, a new and identical cycle begins for each successive D/ω time interval because all subobjects appearing in preceding D/ω interval are completely retrieved during that interval. In the following, we will discuss how to find such set of idle disks, if available, so that this retrieval pattern exists.

When $\omega = 1$, i.e., no common divisor between k and D, $D/\omega = D$. Let im, $0 \le m < M(X)$, be an arbitrarily set of M(X) idle disks chosen at t0. It can be observed that at the end of tD-1, every disk in the system appears exactly once in any of the retrieval sequences Rim(0, D-1), $0 \le m < M(X)$. Meanwhile, all disks will also appear exactly once in each Sf(0, D-1), $0 \le f < M(X)$. Thus, at the end of tD-1, all fragments of the first and only the first D subobjects (i.e., $D \times M(X)$ fragments) are retrieved using arbitrarily chosen M(x) idle disks. Moreover, a new cycle begins for the next and successive D time units because no partially retrieved subobjects from previous D time units are left to be completed in the next D time units.

When $\omega \ge 2$ (obviously, $k \ge 2$), the situation is a bit more complex. For each disk i, designated to serving object X at t0, only D/ω ($< D$) distinct disks appear in its retrieval sequence, i.e., i, $(i + k \times 1) \bmod D$, $(i + k \times 2) \bmod D$, $(i + k \times ((D/\omega - 1))$

mod D, and they are reused every D/ω time units. For example, consider Figure 3.2 again, where k = 2 and D = 14. Since ω = 2, only 7 distinct disks are used in each retrieval sequence. For idle disk 0 chosen at t0, R0(0, 6) consists of only disks 0, 2, 4, 6, 8, 10 and 12, while for disk 1, R1(0, 6) is made of 1, 3, 5, 7, 9, 11, and 13. Similarly, for each disk j storing the f th fragment, $0 \le f < M(X)$, of the 0th subobject, the storage sequence Sf(0, D/ω -1) consists of 7 disks, j, (j + k × 1) mod D, …, and (j + k × (D/ω -1)) mod D.

It can be observed that if i is an idle disk chosen at t0 and it appears in the sequence Sf(0, D/ω) (i.e., j, (j + k × 1) mod D, …, i, (i + k × 1) mod D, …, or (j + k × (D/ω -1)) mod D), then Ri(0, D/ω) contains the same set of D/ω distinct disks as Sf(0, D/ω), except that the order of the disks appearing in the sequences may be different. Thus, if im, $0 \le m < M(X)$, are the idle disks chosen at t0, each of them appearing in a distinct storage sequence Sf(0, D/ω), $0 \le f < M(X)$, then at tD/ω -1, then the same set of disks will appear the same number of times in both Rim(0, D/ω -1), $0 \le m < M(X)$, and Sf(0, D/ω), $0 \le f < M(X)$. That is, the first (and only the first) D/ω subobjects will be completely retrieved during the first D/ω time units using idle disks im, $0 \le m < M(X)$. The pattern repeats for every D/ω time units as no partially retrieved subobjects from previous D/ω interval need to be completed in the next D/ω interval and the set of disks used at t0 is used again at tn×(D/ω), n >0. The following theorem follows.

Theorem. If idle disks chosen to serve an object are each from a distinct storage sequence of the object, then each successive D/ω subobjects from the beginning can be completely retrieved within each successive D/ω time interval.

In the following, we use a simple example to illustrate how idle disks, if available, are chosen to avoid potential disruptions.

Example. Consider a system with 14 disks and a stride k = 4. Assume an request for object X (M(X)=3) has just arrived and currently disks 5, 7, 8, and 10 are idle. We further assume that the fragments of the first subobject of X are stored on disks 2, 3, and 4.

The storage sequences are:
S_0 = 2, 6, 10, 0, 4, 8, 12, 2, 6, …
S_1 = 3, 7, 11, 1, 5, 9, 13, 3, 7, …
S_2 = 4, 8, 12, 2, 6, 10, 0, 4, 6, …

Notice that S0 and S2 are essentially the same sequence. As a result, we can choose disks 8 from S0 (or from S2), 5 from S1, and 10 from S2 (or from S0). Another possible combination could be 8, 7, and 10.

Disruptions can be eliminated taking advantage of these retrieval patterns. Two buffers, each of which can hold D/ω subobjects, may be needed. That is, when the system is displaying previously retrieved D/ω subobjects from one buffer, it retrieves next D/ω subobjects into another buffer. The two buffers are used for input and output alternately.

4.1 Discussions

Disruptions are eliminated using a pair of buffers of size D/ω subobjects each. As conceived, the larger the ω value, the smaller the buffer is required, and the shorter the waiting time (i.e., the time to fill the buffer at the beginning). However, when ω gets larger, it may become more difficult to find suitable idle disks. As readers may have noticed that the disks in the system are in fact divided into ω disjoint groups, each of which has D/ω disks. The ω groups are (i, (i + k) mod D, …, (i + (D/ω -1) × k) mod D), (i +1, (i +1 + k) mod D, …, (i +1+ (D/ω -1) × k) mod D), …, (i + ω-1, (i + ω-1 + k) mod D, …, (i + ω-1 + (D/ω -1) × k) mod D). In order to guarantee the appearance of previously mentioned patterns, idle disks must be chosen from groups corresponding to the storage sequences of the object. Note that more than one disk may need to be chosen from a group when M(X) > ω. The larger the number of groups, the more difficult it is to find a match. Therefore, a trade-off has to be made by the administrator.

In summary, when ω = 1, any arbitrarily chosen M(X) idle disks can be used to display object X. It is most flexible, however, with a bit longer delay of D time units and a bit larger buffer. Note that there is also a delay in the original staggered striping, where only consecutive idle disks are used for display. The delay is the average amount of time waiting for all idle disks to shift to the right positions to read the first subobject. When ω > 1, each idle disk chosen must appears in a distinct storage sequence of the object. It may be more restrictive, but it uses smaller buffers and spends less time in waiting.

5 Conclusions

Bandwidth fragmentation can occur when requests (for objects of varying bandwidths) arrive and are completed. It renders the systems with many small holes of idle disks, which are not large enough to serve requests, and thus can degrades the performance of the system considerably. In this paper, we addressed the problems of potential disruptions in display when fragmented disks are used. We have analyzed the retrieval patterns and identified patterns that can be useful in eliminating disruptions. We have shown that with proper choice of idle disks and a simple buffer scheme, disruptions can be eliminated. Currently, we are investigating further improvement on the waiting time, which is D/ω now.

References

1. Al-Marri, J. A. M., "Variable Bit Rate Continuous Media Servers", USC Computer Science Ph.D. Thesis, Aug. 1998.
2. Berson, S., Ghandeharizadeh, S., Muntz, R, Ju, X., "Staggered Striping in Multimedia Information System", Proc. of the 1994 ACM SIGMOD, vol. 23 no 2, June 1994, pp.79 – 90.
3. Chen,M., Hsiao, H., Li, C., Yu, P., "Using Rotational mirrored Declustering for Replica Placement in a Disk-array-based Video Server", Multimedia Systems, vol. 5, December 1997, pp. 371 – 379.
4. Dashti, A.E., Ghandeharizadeh, S., "On Configuring Hierarchical Storage Structures", Joint NASA/IEEE Mass Storage Conference, March 1998.
5. Freedman, C., DeWitt, D. J., "The SPIFFI Scalable Video-on-Demand System", SIGMOD'95, pp. 352 – 361.
6. Ghandeharizadeh, S., Dashti, A., Shahabi, C., "Pipelining Mechanism to Minimize the Latency Time in Hierarchical Multimedia Storage Managers", Computer Communications, vol. 18, issue 3, March 1995, pp. 170 – 184.
7. Ghandeharizadeh, S., "Stream-based Versus Structured Video Objects: Issues, Solutions, and Challenges", A Book Chapter in Multimedia Database Systems: Issues and Research Directions" Springer Verlag, 1996.
8. Ghandeharizadeh, S., Kim, S. H., "Design of Multi-user Editing Servers for Continuous Media" To appear in the J. of Multimedia Tools and Applications, Kluwer Academic Publishers.
9. Ghandeharizadeh, S., Kim, S. H., Shi, W., Zimmermann. R, "On Minimizing Startup Latency in Scalable continuous Media Servers", Proceedings of Multimedia Computing and Networking conference, Feb. 1997.
10. Ghandeharizadeh, S., Ramos, L., "Continuous Retrieval of multimedia data using parallelism", IEEE Transactions on Knowledge and Data Engineering, vol. 5, no. 4, August 1993, pp. 658 – 669.
11. Gemmel, D. J., Vin, H. M., Kandlur, D. D., Rangan, P.V., Rowe, L. A., "Multimedia Storage Servers: A Tutorial", IEEE Computer, May 1995, pp. 40 – 49.
12. Ghandeharizadeh, S., Zimmermann, R., Shi, W., Rejaie, R., Ierardi, D., Li, T., "Scalable Continuous Media Server", Multimedia Tools and Applications Journal, Kluwer Academic Publishers, vol. 5, issue 1, July 1997, pp.79 – 108.
13. Luther, A.C., "Digital Video in the PC Environment", New York: McGraw-Hill Book Company, 1989.
14. Ng., S., "Advances in Disk Technology: Performance Issues", IEEE Computer, May 1998, pp. 75 – 81.
15. Ozden, B., Rastogi, R., Silberschatz, A., "Periodic Retrieval of Videos from Disk Arrays", IEEE 1997, pp.333 – 343.
16. Raghavan, S. V., Tripathi, S. K., "Networked Multimedia Systems: Concepts, Architecture, and Design", Prentice Hall, 1998.
17. Shahabi, C., "Scheduling the Retrievals of Continuous Media Objects", USC Computer Science Ph.D. Thesis, Aug. 1996.
18. Shi, W., "Data Sharing in Interactive Continuous Media Servers", USC Computer Science Ph.D. Thesis, Aug. 1998.
19. Santos, J. R., Muntz, R., "Design of the RIO (Randomized I/O) Storage Server", Technical Report 970032, Computer Science Dept., UCLA, 1997.
20. Zimmermann, R., "Continuous Media Placement and Scheduling in Heterogeneous Disk Storage Systems", USC Computer Science Ph.D. Thesis, Dec. 1998.
21. Zimmermann, R., Ghandeharizadeh, S., "Continuous Display Using Heterogeneous Disk Subsystems", ACM Multimedia Conference, Nov. 1997.

Subjective Retrieval System for Texture Images Based on Interactions of Orientation and Color

Yuichi Kobayashi[1,2], Toshikazu Kato[2,3]

[1]Toppan Printing Co.,Ltd. Technical Center, Suido 1-3-3, Bunkyo-Ku, Tokyo, Japan
Yuichi.Kobayashi@toppan.co.jp
[2]Electrotechnical Laboratory, Intelligent Systems Division, Human Media Lab, Umezono 1-1- 4, Tsukuba City, Ibaraki Pref. Japan
{y-koba, kato}@etl.go.jp
[3]Chuo University, Dept. of Industrial and Systems Engineering, Kasuga 1-13-27, Bunkyo-Ku, Tokyo, Japan
kato@indsys.chuo-u.ac.jp

Abstract. We have been researching on modeling of visual cognition and its application to content-based image retrieval. We started with analyzing the relationship between pre-attentive vision and attentive vision. And we focused on human early vision as pre-attentive vision. We selected statistical texture images as targets because attentive vision was hard to act them. Previously, many texture features have been proposed, but most of them were insufficient to account for human subjectivity. And so, we have newly designed texture features which were adequate to account for human subjectivity.
From the viewpoint of physiological fundamentals and psychological review, we have focused on the orientation feature and color feature of an image, and calculating contrast of these features in various resolutions. Next, we have measured psychological responses by using some descriptive adjectives, and corresponded them to our texture features by adopting the canonical correlation statistics. Based on this correspondence, we developed a contrast based subjective texture image retrieval (CBSTIR) system. Our system can retrieve images which give a similar impression, and also predict images by some descriptive adjectives. From the experimental results, we found that approximate color property was significant to subjective retrieval, while the precise color distribution was significant to non-subjective retrieval. Moreover, we found that the global orientation interaction in multi resolutions was significant to subjective retrieval.

1 Introduction

With the spreading popularity of computer and network technologies, databases, there has been rapid growth in opportunities for designer with computer, for example, an interior, an exterior design and an industrial design, or for a web page design and so on. With this trend, demand is abruptly growing for how fast and how adequately we can collect texture images we want.

Content-based image retrieval (CBIR) is a key technology to retrieve images adequately. The main problems with CBIR concern what features to extract and how to integrate different features to satisfy user's demands.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 171–182, 2000.

We have been researching about visual impression which means how human beings feel by looking at a scene. We have been studying computational physiology of early visual mechanisms in order to find out important mechanisms for human visual impression[1],[2] and to design adequate features based on the mechanisms.

Texture is important in the appearance of objects in natural scenes, and also a powerful cue in visual perception. Texture analysis and synthesis have been actively researched for the past three decades, and a large number of methods have been proposed.

These methods can be classified into a filtering model and a statistical model. As the former methods, Gabor filtering methods[3],[4], wavelet filter bank methods[5],[6] and Wold model[7] are well known. While, as the latter methods, Gray level co-occurrence matrix (GLCM) methods[8], Markov random field model[9] are well known. All these methods have been well established for how precisely the feature can describe an image. Although the performance of these methods is well known, the demand is rising for the perceptive performance linked to subjectivity. These methods are designed to describe the structure of an image well, and so they are adequate to achieve a structural similarity but not to perceptive similarity.

Recently, not only in computer vision but also in psychology, a general model has been researched. It can describe a wide variety of textures in a common framework, which is consistent with the psychophysical and physiological understanding of human texture perception[10],[11].

In order to achieve perceptive similarity, we started with redesigning features that fit better to human subjectivity.

We hypothesized human subjectivity as a balance of psychological responses (we call this balance an interaction) which caused by some combination of physical features and designed a bottom-up model for human subjectivity. Figure 1 shows the framework of our bottom-up model. Our model focused on color and orientation as extracted features, and calculated contrast of them in multi-resolutions as features. Next, our model calculated the correspondence between each psychological response and each combination of features by canonical correlation statistics.

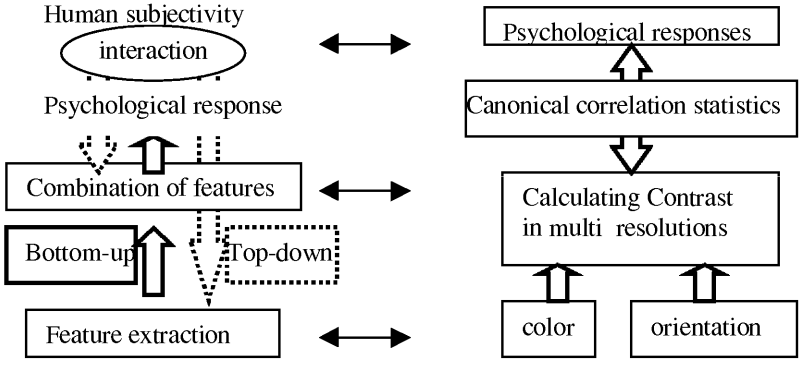


Fig.1. The framework of our CBSTIR model

In this paper, we have focused on human impression, and propose a contrast based subjective texture image retrieval (CBSTIR) model. Section 2 defines the new texture features which are based on color and orientation, and describe a computation of texture features based on the contrast calculation. Section 3 presents psychological responses measured experimentally. Section 4 presents an image retrieval model based on the algorithm that corresponds texture features to the psychological responses. Section 5 presents some experimental results, and Section 6 makes some concluding remarks.

2. Texture feature

In order to focus on a few main physical features, we investigated physiological and psychological researches [11],[12], and visual features of an image that gave clear impressions. In human early vision , there are two kinds of receptive fields. One is to process signal directly, and another is to process the contrast of signals. Motion, color, edge and orientation are processed in early vision.

Then we focused on three basic features for human impression –color, orientation and resolution, locally and globally.

2.1. The local and global color feature

First, we focused on the color feature. In color, both absolute and relative property is important. We focused on color contrast as the relative property and on average color as absolute property. We selected the CIELAB color space, because in the color space, human impression changes linearly and proportionally with color distance. From the CIELAB values, to translate into hue, luminance and saturation is calculated, and then contrast is calculated for each property. Here, we have defined the contrast for each as following:

$$C = \frac{Uppermean(p_i) - Lowermean(p_i)}{mean(p_i)} \tag{1}$$

Here, C is the contrast value and *mean* is the mean value over a local area. *Uppermean* is the mean of values greater than mean value, and *Lowermean* is the mean of values smaller than mean value , over a local area of an image. p_i is each values of hue, luminance and saturation. Contrasts are calculated for each pixel within the neighboring area of the pixel. After calculating contrast over the entire image, the maximum and minimum values are calculated and each value is normalized into the range between them. We experimentally thought average color was better to grasp the global color and contrast was better to grasp local changes of color.

2.2. The local orientation contrast feature

About orientation, there is a physiological fundamental that human early vision has visual cells which reacts selectively only to the orientation of stimulus. Based on this

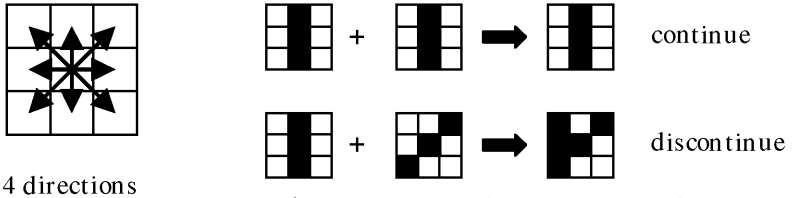
mechanism, we have focused on 2 properties of orientation, one is continuity and another is curvature. In order to evaluate the local continuity or the local discontinuity, we have defined the local orientation contrast feature. The base idea is this: in local area, patterns can be thought as assembly of some line segments. We thought that how these line segments connect or cross is the clue to human impression. If the orientation of the neighbor pixel is the same or similar to that of the fixation point, the orientation continuity is large. On the contrary, if the orientation of the neighbor pixel is perpendicular to that of the fixation point, the orientation discontinuity is maximum. And so, we designed the orientation contrast based on an inner product.

For example, in the case of 3 x 3 pixels area, the contrast between the average pixel value of triplet in 4 direction and the other, and the direction is selected which gives max value, as shown in Fig.2(a). After scanning over an entire image, we can get a local orientation map.

$$OIA_i = 1 - (v_{i-1} * v_i) \tag{2}$$

Here, OIA_i means the orientation contrast value of area i and v_i means the orientation vector of the area i. *(a*b)* means an inner product of a and b. Next, for the local orientation map, we calculate the orientation interaction which is based on an inner product as shown in Fig.2(b).

This measures the strength of continuity or discontinuity to the neighbors.



(a) 4 directions to select triplet

(b) continuity or discontinuity with a neighbor

Fig.2. The diagram of calculating the orientation contrast of 3x3 pixels area.

Next, in order to grasp the global distribution of each local orientation feature, we introduced the local autocorrelation feature[13], and the n-order correlation is calculated as follows:

$$cor(a_1, a_2, \Lambda\ , a_N) = \int v(r)v(r+a_1)v(r+a_2)\Lambda\ v(r+a_N)dr \tag{3}$$

Here, a_i means the shift from the fixation point and r means the position of the fixation point.

We adopted up to the second order correlation, in order to grasp the curvature. This is also because their kernels mimicked some receptive fields in the early vision that responded selectively for each orientation[12]. This method has also the advantage of being shift invariant and computationally inexpensive. Our model

computes 25 local autocorrelation coefficients from a digital image, using the kernels represented in Fig.3. Each kernel is scanned over the entire image, and for each possible position, the product of the pixels marked in black is computed. All the products corresponding to a kernel are then summed so as to provide one coefficient. By dividing the auto-correlation coefficient by powers of the zero-order coefficient, optional value invariance may be implemented.
By this feature, we get 25 dimensional vector for one property of an image with one resolution.
We applied these computation for 3 properties of hue, luminance and saturation, and so 75 dimensional vector with one resolution.

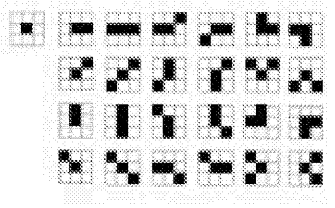


Fig.3. Set of auto-correlation kernels.

2.3. Resolution

At last, the resolution with which people see an image is also important to human impression, which means how roughly or finely people see an image. We adopted three levels of resolution of an image. For 256 x 256 pixel-sized images, a half, a quarter and an octant resolution of an original image are selected as a result of some trials. We included the high resolution, the middle resolution and the low resolution to each.

We computed the local orientation features for 3 color properties (hue, luminance and saturation) and in these 3 resolution levels, and then we get a 225 dimensional orientation vector for an image.

Finally, by synthesizing the absolute color features and the local orientation features, we defined the 228 dimensional vector as a new texture feature.

3. Psychological responses

In order to measure human impression for texture images, we took some psychological experiments for some texture images oriented for interior design. As well as an conventional approach, we measured human impression by psychological experiment (Semantic Differential method). Lohse,et al researched human texture perception by psychological experiments[14]. By taking their method into account, we designed the experiment that we prepared some adjectives and had subjects answer the extent to how they feel for each adjective.

3.1. Items

In order to measure subject's impression, we used several digital images of texture and several adjectives. All the texture images were selected from royalty free CD-ROMs of material such as wood, paper, cloth, stone, metal and so on. All the adjectives were selected from the dictionary, and the discussion with some interior designers. At last, 22 pairs of adjectives and 180 texture images were selected experimentally. The 22 pairs of adjectives are shown in Table.1.

Table 1. Adjective words used in experiments.

Warm/ cool	Hard/ soft	Sharp/ dull	Dynamic/ calm	Elegant/ unpolished
Wild/ subdued	Casual/ Formal	New/ old	Modern/ Old-fashioned	bright/ dark
Refresh /gloomy	Rough/ Smooth	Heavy/ light	showy/ sober	organized/ random
Clear/ Unclear	Wet/ Dry	Open/ confined	Tense/ relax	Interesting/ boring
glossy/ mat	Rich/ simple			

3.2. Subjects

A hundred non-designer subjects and a hundred designer subjects participated in the experiment. The background of the subjects was unified with respect to age, level of education and ethnic origin, and varied with respect to the area of specialization. Half of the subjects were female and the other were male. All non-designer subjects were in 20's employee in Tokyo area. Half of them worked in a technical session and the other worked in non-technical session. All designer subjects were students around 20 who major in design in Tokyo area.

3.3. Tasks

We designed the two types of experiments. In one experiment, we had subjects classify texture images into some similar groups and give adjectives which can describe the impression to each group. The another is to measure the human subjective impression for various texture images by the semantic differential(SD) method [15]. Subjects watched each texture image presented on display, and gave score to each adjective pairs. Fig.4 shows a pairwise scale.

We did these experiments in our laboratory which luminance was set to that of the daytime office.

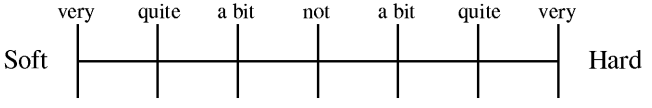


Fig.4. The example of SD-scale.

4. Subjective Image Retrieval System

4.1. Canonical correlation statistics

In order to correspond psychological responses to our texture features at one time, we adopted a canonical correlation statistics [16]. And we developed a subjective retrieval system of texture images. First, a factoring analysis is applied to both texture features and psychological responses, respectively, and principal vectors are calculated. Next, canonical correlation is applied to these principal vectors, and we get canonical coefficients to correspond them. These coefficients can be used to project a texture feature vector to a subjective response vector, or vice versa.

4.2. Similarity measure

For our texture features, both distance similarity and direction similarity are needed, and so we have designed the similarity measure as follows:
Here,v_1 and v_2 are vectors compared with, * means an inner product, ||v|| means norm

$$Sim. = 1 - \frac{(v_1 * v_2)}{\|v_1\|\|v_2\|} + 1 - \frac{1}{1 + \|\|v_1\| - \|v_2\|\|} \tag{4}$$

of vector v and |v| means the absolute value of v.
We decided similarity order by means of this similarity measure.

4.3. Image retrieval

Our system enables a subjective image-to-image retrieval by means of comparing projected vectors of our texture features by using canonical correlation coefficients.

Moreover, our system enables to retrieve images subjectively. Users only gives his feeling by weighting some descriptive adjective words selected from prepared 22 pairs of adjective words, and the weighting value is selected from 1 to 7 according to what extent they feel.

5. Experimental Results and discussion

5.1. Comparing clustering results

In order to check the accordance with subjectivity, we compared clustering results. For comparison, we adopted the correlation features based on GLCM[8], because they were well known to grasp global and local physical characteristics of an image. The results are shown in Table.3. We adopted 8 cluster level as the comparison level. Here, fitness means the degree of fitting between the psychological result and each feature's result. For stone textures, our local orientation interaction features could much more finely cluster than the other features, and fit much better to the result by psychological responses. While, our local orientation features could cluster better for wood and cloth textures than the other.

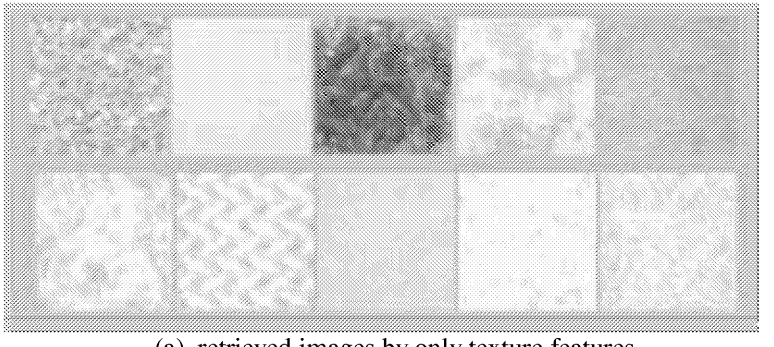
Table 3. The comparison of clustering results.

	Psychological response	Orientation interaction feature	Local contrast feature	Correlation based on GLCM	
wood	14	22	12	7	Cluster number
	-	11	9	17	Fitness
cloth	11	11	10	7	Cluster number
	-	15	14	11	Fitness
stone	11	10	7	3	Cluster number
	-	17.5	11.5	-	Fitness

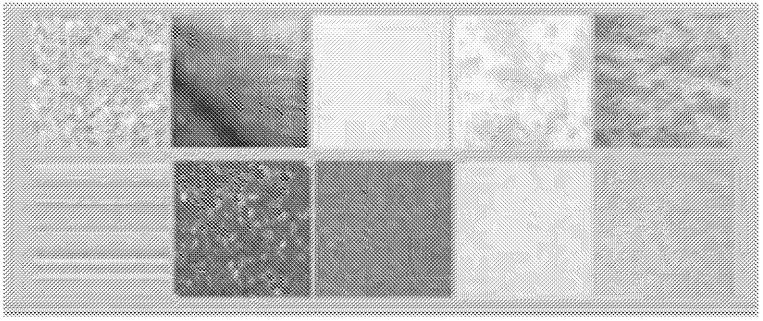
5.2. Image-to-image retrieval

Our system can retrieve images both by comparing feature vectors of images and by comparing projected vectors of feature vectors using psychological experimental results.

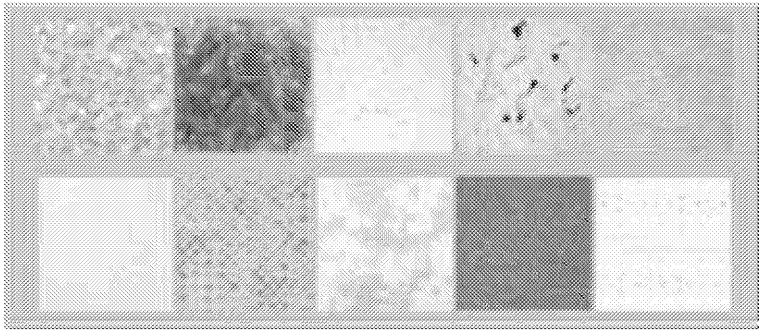
Fig.5. show some examples of our image-to-image retrieval performance. Fig.5(a) show the retrieved images by comparing feature vectors. Fig.5(b) and (c) show the retrieved image by comparing projected vectors. (b) uses the projection of non-designer subjects who didn't major in design and don't work at designing division. (c) uses the projection of students subjecting design. The first image of each results is the key image. Fig.5(a) involves various kinds of texture images, and they are similar to the key image in some pattern, but the impression for them is various. Fig.5(b) involves images which are different from key image in pattern and color, but have some similar impression. From these results, the results using the projection of designer subjects seem to give more similar impression than using the projection of non-designer subjects.



(a). retrieved images by only texture features



(b). retrieved images by using projection (non-designer)



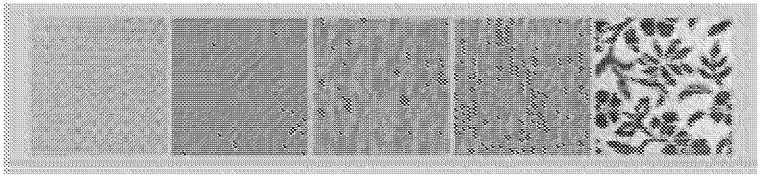
(c). Retrieved images by using projection(designer)

Fig.5. Image retrieval of mixed domain textures: (a) comparing feature vector, (b) comparing projected vectors by non-designer subjects (c) comparing projected vectors by designer subjects

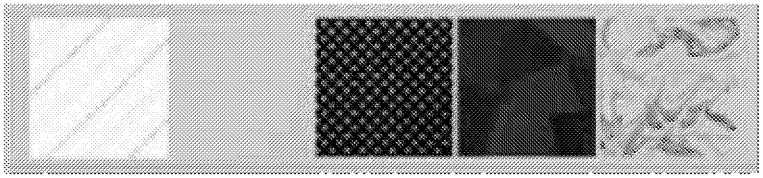
5.3. Adjectives-to-image retrieval

In Our system, the principal component analysis was applied to the psychological responses, and they were degenerated to 6 principal component axes, and these 6 axes are correlated to the texture features.

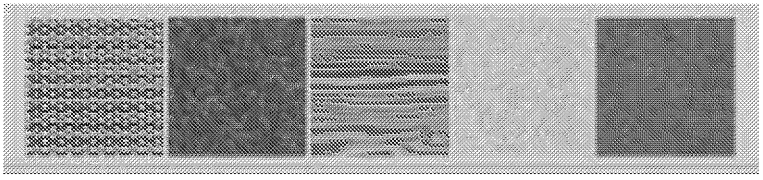
Fig.6 shows the top 5 texture images retrieved by giving weights to some adjectives. For each case, both results are shown, one is retrieved by using projection of non-designer subjects and another is by that of designer subjects. The results show that our system can retrieve texture images according to human subjectivity. While, as you can see, the results differ very much between these two populations of subjects. They also show that the retrieved results depend much on what population to be selected for calculating projection. Therefore, better selection of population will match better to the population's subjectivity. We think that for subjective retrieval, it is very important to select subjects in accordance with the characteristics of users.



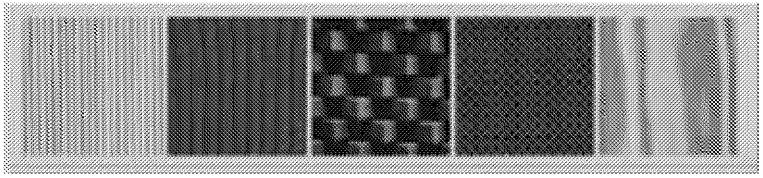
(a). Elegant (by population of non-designer subjects)



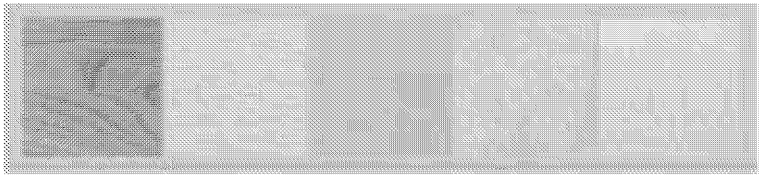
(b). Elegant (by population of designer subjects)



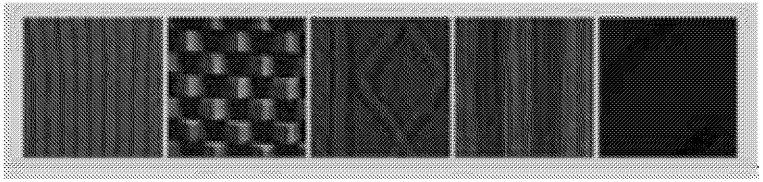
(c). Organized (by population of non-designer subjects)



(d). Organized (by population of designer subjects)



(e).Warm, Soft, Unclear (by population of non-designer subjects)



(f). Warm, Soft, Unclear (by population of designer subjects)

Fig.6. The examples of results of adjective-to-image retrieval of our system.

6. Concluding Remarks

We proposed a contrast based subjective texture image retrieval (CBSTIR) model. Our model is based on calculating contrast for color and orientation, globally and locally, with various resolution. CBSTIR model corresponds our texture features to psychological responses by canonical correlation statistics. We have tried various kinds of texture features which have been proposed so far. Our texture features can classify texture images to fit to human subjectivity better than other texture features

As the results of adapting our model to stone, wood and cloth texture images, we found that discontinuity of local orientation is significant for stone images, and contrast of local orientation is significant for cloth and wood images. Only for wood images, correlation feature was better, because wood images were not so variable in color and patterns.

We have developed a contrast based subjective texture image retrieval (CBSTIR) system. CBSTIR system can retrieve texture images subjectively in both image-to-image retrieval and adjectives-to-image retrieval by giving weights to some adjectives. In each case, subjective retrieval can be done with either of the projection of two populations – non-designer and designer subjects. As the results of comparing their retrieval, the results differ more in adjectives-to-image retrieval than in image-to-image retrieval. We think this is because dependency on linguistic logic in human subjectivity is higher in population of non-designer than in population of designer.

CBSTIR model takes a bottom-up approach based on the relationship between processing of primitive signals such as orientation and color and psychological responses. In human subjectivity, another level of processing is also important, which is the symbolic processing. Here, symbol means some kind of knowledge which is composed of combination of some features, such as stereotypical impression for material, structure, shape and so on. For example, “cool”, “hard” for iron, “warm”, “relaxed”, “hard” for wood and “sharp”, “tense” for triangle. This type of processing belongs to a top-down approach, or attentive vision problem. In order to improve performance of subjective retrieval, the model which can deal with this type of processing will be needed.

We conclude that contrast of color and orientation with multi-resolutions is essential for human subjectivity. We found that an absolute color property is essential globally, while orientation is essential locally, to human subjectivity.

We also found that the local autocorrelation feature was an effective measure to grasp the global orientation from each local orientation properties into one global property.

Our future work is to plan to evaluate our model by evaluation experiments, and to try to establish model which controls each texture features according to subject’s attention, and discriminate the material of a visual target, for the ultimate aim of understanding human subjectivity.

7. Acknowledgment

This research is a part of “Human Media Project” funded by New Energy and Industrial Technology Development Organaization(NEDO).

References

1. T.Kato,T.Sakamoto, Y.Kobayashi, "Mathematical model of lateral inhibition and light/dark adaptation mechanisms", Technical Report of IEICE, PRMU96-72, 1996, pp.33-40.
2. Y.Kobayashi, T.Kato, "A High Fidelity Contrast Improving Model Based on Human Vision Mechanisms", Proc. of IEEE ICMCS'99, Vol.2, June 1999, pp. 578-584.
3. J.G.Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters", J.Opt.Soc.Am.A, Vol.2, No.7, July 1985, pp. 1160-1169.
4. M.R.Turner, "Texture Discrimination by Gabor Functions", Biol.Cybern., 55, Springer-verlag, 1986, pp. 71-82.
5. M.Unser, "Texture Classification and Segmentation Using Wavelet Frames", Trans. on Image Processing, Vol.4, No.11, November 1995, pp. 1549-1560.
6. T.S.Lee, "Image Representation Using 2D Gabor Wavelets", Trans. on Pattern Analysis and Machine Intelligence, Vol.18, No.10, October 1996, pp.959-971.
7. F.Liu, W.Picard, "Periodicity,Directionality,and Randomness: Wold Features for Image Modeling and Retrieval", Trans.on Pattern Analysis and Machine Intelligence, Vol.18, No.7, July 1996, pp.722-733.
8. R.W.Conners, C.A.Harlow, "A Theoretical Comparison of Texture Algorithms", Trans. on Pattern Analysis and Machine Intelligence, Vol.2, No.3, May 1980, pp.204-222.
9. R.Chellappa, S.Chatterjee, "Classification of Textures Using Gaussian Markov Random Fields", Trans. on Acoustics,Speech, and Signal Processing, Vol.33, No.4, August 1985.
10. B.Julesz, "Textons, the elements of texture perception and their interactions", Nature, 290, 1981, pp.1619-1645.
11. A.Treisman,S.Gormican, "Feature Analysis in Early Vision: Evidence From Search Asymmetries", Psychological Review, Vol.95,No.1,1988,pp.15-48.
12. M.S.Livingstone, D.H.Hubel, "Psychophysical evidence for separate channels for the perception of form, color, movement, and depth.", Journal of Neuroscience, Vol.7, No.11, 1987,pp.3416-3468.
13. N.Otsu, T.Kurita, "A New Scheme for Practical, Flexible and Intelligent Vision Systems", Proc. of 11th ICPR, 1992, pp.213-216.
14. A.R.Rao, G.L.Lohse, "Identifying High Level Features of Texture Perception", CVGIP, Vol.55, No.3, 1993, pp.218-233.
15. C.E.Osgood, G.J.Suci, P.H.Tannenbaum, The Measurement of Meaning, University of Illinois, Urbana, 1957.
16. T.Kurita, A Study on Applications of Statistical Methods to Flexible Information Processing", Researches of the Electrotechnical Laboratory, Japan, 1993.

Chasing Relational Database Constraints Backwards

Stéphane Coulondre

LIRMM (CNRS/Université Montpellier II)
161 rue Ada, Montpellier Cedex 5, France
coulondr@lirmm.fr

Abstract. Data dependencies are well known in the context of relational database. They aim to specify constraints that the data must satisfy to model correctly the part of the world under consideration. The implication problem for dependencies is to decide whether a given dependency is logically implied by a given set of dependencies. A proof procedure for the implication problem, called "chase", has already been studied in the generalized case of tuple-generating and equality-generating dependencies. The chase is a bottom-up procedure: from hypotheses to conclusion, and thus is not goal-directed. It also requires the dynamic creation of new symbols. This paper introduces a new proof procedure which is top-down: from conclusion to hypothesis, that is goal-directed. The originality of this procedure is that it does not act as classical theorem proving procedures, by requiring a special form of expressions, such as clausal form, obtained after skolemisation. We show, with our procedure, that this step is useless, and that the notion of *piece* allows inferring directly on dependencies, without dynamically creating new symbols. With the recent introduction of constrained dependencies, some interesting perspectives also arise.

1 Introduction

Dependency theory allows the expression and modelling of constraints that the data must satisfy in order to reflect correctly the world that a relational database intends to describe. Since the introduction of functional dependencies by Codd ([Cod72]), many kinds of dependencies have been studied in the litterature, and a lot of work has been carried out in the late 70's and early 80's. Database dependencies theory is still an active area of research [SF00] [Her95] [LL97a] [LL97b] [LL98]. Functional and multivalued dependencies are the most known classes of data dependencies. In practice, these two kinds are sufficient in general to express constraints ([Ull88]). Nevertheless, more general classes have been introduced, with the purpose of finding a uniform way to express constraints ([BV81],[SU82]). This paper deals with the class commonly known for generalizing most of the dependencies: that of tuple-generating and equality-generating dependencies (TGDs and EGDs) ([BV84b]). For a survey on this general class of dependencies, we refer the reader to [FV86].

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 183–192, 2000.

The central problem is the implication problem, which is to decide whether a dependency is logically implied by a given set of dependencies. A process that could solve this problem would provide a solution to find the minimal cover of a set of dependencies, to decide whether a dependency is redundant within a set, useful during the constraint acquisition stage, etc. A procedure has already been designed in [BV84b] for that purpose: the well-known *chase*. Unfortunately, as the implication problem for TGDs and EGDs is semi-decidable, the *chase* is only a proof procedure, and therefore the process may run forever. As we argue in this paper, the *chase* is clearly a bottom-up procedure: from hypotheses to conclusion. Also, the *chase* requires the dynamic creation of new symbols.

We introduce a new proof procedure which is top-down: from conclusion to hypothesis that is goal-directed. The originality of this procedure is that it does not act as classical theorem proving procedures, by requiring a special form of expressions, such as clausal form, obtained after skolemisation. We show, with our procedure, that this step is useless, and that the notion of *piece* allows inferring directly on dependencies without dynamically creating new symbols.

Therefore, our top-down chase is not simply the usual chase reversed, but a new way of solving the implication problem. The fact it can be performed top-down is the first contribution of this paper. The second contribution is to avoid dynamic creation of symbols, as well as skolemization usually applied on the original formulae prior to top-down proofs. This is realized by taking advantage of the form of the dependencies. To our knowledge, this has not been realized before. Indeed, dynamic creation of symbols can be a costly operation in general proof procedures, such as the *chase*, in order to take into account the effect of existential quantifiers.

While it is true that top-down approaches can take exponential time longer w.r.t. bottom-up approaches, many reasons allow us to think that the proof procedure presented in this paper can be efficient. These arguments will be detailed hereinafter. The improvement is currently being formally assessed.

Recently, constrained dependencies have been introduced ([Mah94], [BCW95], [Mah97]). They originate in the constraint programming field and permit expression of semantic relations on variables, thus giving them an interpretation. The chase procedure has been redesigned in [MS96], still in a bottom-up way, in order to deal with constrained tuple-generating dependencies. This work in the dependency theory gives new perspectives for the top-down chase procedure we present.

The top-down chase originates in the conceptual graphs model, which is a knowledge representation model introduced by Sowa ([Sow84]). The base model has been extended with graph rules and an inference method, called *piece resolution* ([SM96]). The logical roots of this process have been studied in [CS98], and constitute the basis of the top-down chase. Proofs of the lemmas and theorems of this paper are therefore derived from these two last-mentioned.

Section 2 describes the framework and the implication problem for data dependencies. We sketch the existing (bottom-up) chase. In section 3 the top-down chase is explained. Section 4 closes the paper with some concluding remarks. For

sake of room, we cannot formally present the proposed mechanism, which can be found in [Cou00].

2 The Framework

The first subsection states some assumptions we make throughout this paper and presents the kind of dependencies this paper focuses on. Note that they are known to capture the semantic of most of the dependency types studied in the litterature. The second subsection presents the implication problem and then describes the traditional chase procedure, which has been designed to solve the implication problem.

2.1 Preliminaries

TGDs and EGDs can all be expressed in first-order logic. An *equality-generating dependency* (EGD) says that if some tuples exist in the database, then some values in these tuples must be equal. A *tuple-generating dependency* (TGD) says that if some tuples exist in the database, then some other tuples, whose values are not necessarily taken from the first tuples, must also exist in the database. Thus some values may be unknown.

For example, to express a classical functional dependency stating that two customers having the same name also have the same identifier, we use the following EGD which is also a functional dependency :

$$\forall custid_1\ \forall custid_2\ \forall custname\ (cust(custid_1, custname) \wedge cust(custid_2, custname) \rightarrow custid_1 = custid_2)$$

To express that a invoice is always related to an existing order, we use the following TGD :

$$\forall invoiceid\ \forall amount\ \forall orderid\ (invoice(invoiceid,\ amount,\ orderid) \rightarrow \exists custid\ order(custid,\ orderid))$$

By introducing constants, we can state some more specialized constraints. For example, the invoice 23 is related to an order taken by the customer 12 :

$$\forall amount\ \forall orderid\ (invoice(23,\ amount,\ orderid) \rightarrow order(12,\ orderid))$$

Throughout this paper, we assume that the universal relation assumption holds, i.e. the database is modelled with only one relation. While this assumption is very unpractical for real-world applications, it usually permits a simpler presentation of the approaches. Secondly, dependencies are typed (many-sorted), so attribute domains are disjoint. These restrictions appear here for clarity reasons only and all the results are applicable in the unrestricted model.

2.2 The implication problem and the (bottom-up) chase

Let D be a set of dependencies, and d be a dependency. The implication problem is to decide whether $D \models d$, that is to determine whether d is true in every database in which each dependency of D is true. Let $SAT(D)$ be the set of all relations that satisfy all the dependencies in D, then the implication problem is equivalent to decide whether $SAT(D) \subseteq SAT(d)$.

The chase procedure has been designed to solve the implication problem. The reader can refer to [BV84b] for a complete description. Informally speaking, if the dependency to be proven is a TGD with hypothesis I and conclusion I' or a EGD with hypothesis I and conclusion $x = y$, the chase procedure assigns a symbol to each variable occuring in the dependency (note that this is not a valuation, but just a bijection with a set of symbols). Then it takes the hypothesis, and treats it as if it formed a set of tuples. Then it applies repeatedly the dependencies of D, following two distinct rules, one for TGDs, whose effect is to add tuples to the relation, and one for EGDs, whose effect is to "identify" two symbols. When the dependency to be proven is a TGD, the procedure stops when obtaining the tuples of I' (modulo renaming of symbols assigned to existential variables of I'). When it is a EGD, the procedure stops when obtaining the identification of x and y.

This mechanism has been shown to be sound and complete in [Var84]. Note that the implication problem for TGDs and EGDs is semi-decidable. Thus the chase may not stop.

The chase procedure is clearly a bottom-up (or forward chaining) one. Indeed, rule applications are generating new tuples or identification of symbols. This is executed until we obtain the desired conclusion. The goal to be proven is not used to guide the process. Moreover, when applying a TGD rule whose effect is to add tuples to the relation, existential quantification always requires a costly dynamic creation of new symbols.

We now show that we can apply a top-down (or backward) procedure, in which the process is goal-directed, that does not require dynamic creation of symbols.

3 The Top-Down Chase

3.1 Preliminaries

Depending on the type of the dependencies, the implication problem is solvable or recursively unsolvable ([BV81,Var84,GL82]). This means that in the first case, there is a decision procedure, hence an algorithm that always halt, whereas in the second case, there is a proof procedure: if the implication is true, then the process will terminate. In the other case, it might never stop.

A decidable subset of TGDs is known as Full TGDs. These dependencies have the same form as Datalog rules. In this case, the notion of *piece*, stated in the introduction, is not applicable. Therefore, many theorem proving procedures

can be applied to solve the implication problem involving only Full TGDs, such as Query-Subquery ([Vie86]), OLD-Resolution ([TS86]) and XSB ([SSW94]).

For this kind of dependencies, as well as all other subsets (functional and multivalued dependencies) the interest of the top-down chase is to be also applicable. However, in these particular cases, specific proof procedures shall be more efficient. The top-down chase, as well as the classical bottom-up one, are clearly needed when dependencies are more complex (i.e. not a decidable subset), and also to provide a general way to solve the implication problem.

3.2 A top-down proof procedure

First, we focus on TGDs only. EGDs will be taken into account later.

Let D be a set of TGDs, and let d be a TGD with hypothesis I and conclusion I'. Intuitively, to decide whether $D \models d$, we also, as for the chase, assign a new symbol to each variable occuring in the dependency. We only make the assumption that the symbols assigned to the variables in the hypothesis are a special subset of the symbols used, called constant symbols.

Then we take the conclusion, and treats it as if it formed a set of tuples. Let Q be this relation. Q is considered as the *goal* to reach. On the other hand, we add the hypothesis to D, by transforming each tuple into a TGD that has no variables not hypothesis and therefore is a *fact*. The result is noted D_ω.

Example 1. Let $d = \forall x \forall y (r(a,b,x) \wedge r(a,y,c) \rightarrow \exists z \exists w (r(z,y,x) \wedge r(z,e,f) \wedge r(w,b,f)))$ the dependency to be proven. Let h the following symbol mapping : $x \mapsto x_1$, $y \mapsto y_1$, $z \mapsto z$, $w \mapsto w$ from which only x_1, y_1 are constant symbols. Then Q is the relation formed by the tuples $r(z, y_1, x_1)$, $r(z,e,f)$ and $r(w,b,f)$, and $D_\omega = D \cup \{r(a,b,x_1)\ ,\ r(a,y_1,c)\}$.

The main process is described as follows: we try to remove the tuples of Q, by successively applying a rule according to the TGDs of D_ω, giving each time a new goal. These rule applications may introduce new tuples in Q. If we achieve in removing all tuples, i.e. obtaining an empty goal, then $D \models d$.

We now describe the main step of the process. Given a TGD and a goal, this rule constructs a new goal. We need to introduce the notion of piece by a constructive definition.

Definition 1 (Piece). *Given a set of variables V, a piece is a set of atoms that are linked by at least a variable of V. Each variable can appear only within one piece. Two atoms that does not share any variable of V are not in the same piece.*

Example 2. Let $V = \{u, v\}$ be a set of variables, and let $C = \{r(a,b,t), r(a,u,c), r(v,u,t), r(v,e,f)\}$ be a set of atoms. There are two pieces of C w.r.t. V that are $\{r(a,u,c),\ r(v,u,t),\ r(v,e,f)\}$ because they share u and v and $\{r(a,b,t)$ that does not share any variable of V with the other atoms.

When applying the core rule, pieces are searched within the goal. The atoms of each piece can be treated at the same time. This notion is an alternative to the classical skolemisation process that replaces unknown values by constants in order to break formulae in atoms while keeping information about the range of existentially quantified variables, prior to using classical first-order logic provers. A piece is a set of atoms which share unknown values, and are therefore treated as a whole. This is the originality of the top-down chase.

We now describe the core rule.

Definition 2 (Core rule). *Given the goal Q and a TGD T which do not have any symbol in common, except constants and constant symbols : if we can find a symbol assignment of the universally quantified variables V of T, and a renaming of the non-constant symbols of Q such that a piece of Q, defined w.r.t. the existentially quantified variables of T, appears in the conclusion of T, then the new goal is obtained by removing the piece from Q and by adding the hypothesis of T.*

Example 3. Let $T = \forall t \forall u(r(a,e,t) \wedge r(d,u,t) \rightarrow \exists v(r(a,b,t) \wedge r(a,u,c) \wedge r(v,u,t) \wedge r(v,e,f)))$. Let the symbol assignment of the universally quantified variables of $T : t \mapsto x_1$, $u \mapsto y_1$, and a renaming of the non-constant goal symbols : $z \mapsto v$ and $w \mapsto w$. After symbol assignment, the hypothesis of T is composed of the following atoms : $r(a,e,x_1)$ and $r(d,y_1,x_1)$ and the conclusion is composed of $r(a,b,x_1)$, $r(a,y_1,c)$, $r(v,y_1,x_1)$ and $r(v,e,f)$. After renaming of non-constant symbols, the goal Q is composed of $r(v,y_1,x_1)$, $r(v,e,f)$ and $r(w,b,f)$. The pieces of the goal, defined w.r.t. $\{v\}$ are $\{r(v,y_1,x_1), r(v,e,f)\}$ and $\{r(w,b,f)\}$. The first piece appears in the conclusion of T, therefore, we can construct the new goal by removing it and adding the hypothesis : Q is now modified and formed by the tuples $r(a,e,x_1)$, $r(d,y_1,x_1)$ and $r(w,b,f)$.

Note that there may be several possible TGDs obtained by an application of the rule, depending on the symbol assignment and the symbol renaming.

Theorem 1. *There is a sequence of rule applications such that the sequence starts with the original goal, and gives the empty goal as an end result if and only if $D \models d$.*

EGDs add some difficulties because they include an equality predicate. Nevertheless, it has been shown in [BV84b] that the implication problem for TGDs and EGDs is reducible to the implication problem for TGDs. Thus, our core rule is sufficient.

4 Conclusion and remarks

As a conclusion, we discuss several points. First we compare the top-down chase with a backward formal system and with other proof procedures. Then we discuss the need of implementation strategies and the contributions of our work. We conclude this paper by lifting some restrictions on the model and pointing out some perspectives.

4.1 Comparison with formal systems

Many formal systems have been studied for data dependencies ([BFH77], [Sci82], [BV84a], [SU82]). In [BV84a], some formal systems for TGDs and EGDs are studied. Two of these systems are backward, but only one, namely the T3 formal system, has some similarities with the top-down chase. We sketch here the main differences. We refer the reader to this paper for more details about formal systems for TGDs and EGDs.

The T3 system deals with TGDs and their existentially quantified variables in the following way : the process starts with a goal tuple Q and applies a TGD T by making the goal and the conclusion of T coincide. Typically this is achieved using especially the collapsing, augmentation and projection rules. When they coincide, it uses the transitivity rule to derive a new goal. The derivation scheme forms a tree and is not linear, as it is the case for SLD-resolution.

The top-down chase leads to a linear inference in the sense that it uses directly TGDs from the base without first applying rules on them. There is only one rule. Obviously, it is a more complex step.

4.2 Implementation strategies

To implement the top-down chase, we need to add a search strategy (breadth-first or depth-first for example) that obviously may not terminate. From a first-order logic point of view, the whole principle can be seen very intuitively as a kind of SLD-resolution, dealing with groups of atoms instead of only one at the time. So far, we have not studied any optimization for the top-down chase, which should gain a lot from traditional top-down optimization techniques.

4.3 Related work

To our knowledge, this is the first time a top-down proof procedure is used to solve the implication problem for dependencies. As such, a first constribution is to have shown that this can be performed top-down. The top-down chase avoids the skolemisation process, necessary in order to use classical top-down theorem proving procedures. Indeed, this step can lead to exponential increase of the size of the formula. By providing a way to infer directly on the original form of dependencies, by way of the notion of piece, the top-down chase is conceptually speaking a simpler approach.

Compared to the classical chase, it does not require the dynamic creation of new symbols. Only the preliminary transformation of the TGD to be proven requires creating as many new symbols as universally quantified variables appearing in it. Thus this step is unsignificant. This is our second contribution.

Note that the top-down chase is not the usual chase simply reversed. The core rule is totally different from those used in the bottom-up approach of [BV84b].

We are currently formally investigating the efficiency of the top-down chase. Actually, this is not trivial to assess, because of the various parameters that come into play. We shall detail some of them. First of all, there is no need for

skolemization nor dynamic symbol creation. This might increase the efficiency. On the other hand, top-down approaches can take exponential time longer w.r.t. bottom-up approaches. However, the notion of piece performs some dynamic optimizations by dealing with groups of atoms instead of one at the time, thus failures are detected earlier w.r.t. classical top-down procedures. But this gain is hard to assess, though it sometimes dramatically reduces the number of backtracks. Moreover, in worst cases, the resolution tree can never be larger. For example in the SLD-Resolution, a branch would be developped and the failure would appear later. As an illustration, we have compared our method with Prolog over 5000 dependencies generated at random. The top-down chase has been implemented on top of the CoGITo platform [GS98], which is a set of tools for conceptual graphs management. These tests showed that the top-down chase provides a marked average improvement for the number of backtracks as well as for the proof duration (when the process stops), in spite of a non-optimized implementation. A detailed illustrated analysis can be found in [Sal97]. However, we must notice that the core rule has an exponential complexity in order to find the right symbol mappings.

For all these reasons, we plan to implement the chase of [BV84b] in order to practically compare their efficiency. Dependencies might be divided into classes for which one or the other approach might be better.

We must mention some optimisations to the bottom-up approach have been made by magic-sets in [MS96]. The principle of magic sets ([BMSU86]) is to perform at compile time some optimizations that are usually performed at runtime, by rewriting the set of dependencies before inference. This leads to avoiding the generation of irrelevant facts during the process, which is the essence of the top-down approach.

4.4 Extension of the model

As already stated, we assumed some restrictions on the model that can be easily lifted. The reduction of EGDs to TGDs also works in this unrestricted model, which is stated in [BV84b]. Thus all the results can be extended, as they are in [CS98] for piece resolution.

There have been recent advances in data dependency theory with the introduction of constrained dependencies. This type of dependency can express a wide variety of constraints on the data ([BCW95]), besides generalizing most of the temporal dependencies of the taxonomy presented in [JS92]. The chase procedure has been redesigned, still in a bottom-up way, in order to deal with constrained tuple-generating dependencies ([MS96]), which are constrained functional dependencies. Our procedure can serve as a basis for the design of a top-down chase for constrained tuple-generating dependencies.

Acknowledgments. The author is grateful to Moshe Y. Vardi for useful comments and to some anonymous referees for their suggestions.

References

[BCW95] Marianne Baudinet, Jan Chomicki, and Pierre Wolper. Constraint-generating dependencies. In Georg Gottlob and Moshe Y. Vardi, editors, *Database Theory—ICDT'95, 5th International Conference*, volume 893 of *Lecture Notes in Computer Science*, pages 322–337, Prague, Czech Republic, 11–13 January 1995. Springer.

[BFH77] Catriel Beeri, Ronald Fagin, and John H. Horward. A complete axiomatization for functional and multivalued dependencies in database relations. In Diane C. P. Smith, editor, *Proceedings of the 1977 ACM SIGMOD International Conference on Management of Data*, pages 47–61, Toronto, Canada, 3–5 August 1977. ACM, New York.

[BMSU86] François Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman. Magic sets and other strange ways to implement logic programs. In *Proceedings of the Fifth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 1–16, Cambridge, Massachusetts, 24–26 Marseille 1986.

[BV81] Catriel Beeri and Moshe Y. Vardi. The implication problem for data dependencies. In Shimon Even and Oded Kariv, editors, *Automata, Languages and Programming, 8th Colloquium*, volume 115 of *Lecture Notes in Computer Science*, pages 73–85, Acre (Akko), Israel, 13–17 July 1981. Springer-Verlag.

[BV84a] C. Beeri and M. Y. Vardi. Formal systems for tuple and equality generating dependencies. *SIAM Journal on Computing*, 13(1):76–98, 1984.

[BV84b] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, October 1984.

[Cod72] E. F. Codd. *Further Normalization of the Database Relational Model.* R. Rustin, Ed. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[Cou00] Stphane Coulondre. A top-down proof procedure for data dependencies. Technical report, LIRMM, Montpellier, France, 2000.

[CS98] Stéphane Coulondre and Eric Salvat. Piece resolution: Towards larger perspectives. In *Proceedings of the Sixth International Conference on Conceptual Structures (ICCS-98)*, Montpellier, France, 1998.

[FV86] R. Fagin and M. Y. Vardi. The theory of data dependencies: a survey. In *Mathematics of Information Processing, Proceedings of Symposia in Applied Mathematics*, volume 34, pages 19–72. American Mathematical Society, 1986.

[GL82] Yuri Gurevich and Harry R. Lewis. The inference problem for template dependencies. *Information and Control*, 55(1–3):69–79, October/November/December 1982.

[GS98] David Genest and Eric Salvat. A platform allowing typed nested graphs: How cogito became cogitant. In *Proceedings of the Sixth International Conference on Conceptual Structures (ICCS-98)*, LNAI, Berlin, 1998. Springer-Verlag.

[Her95] Christian Herrmann. On the undecidability of implications between embedded multivalued database dependencies. *Information and Computation*, 122(2):221–235, 1 November 1995.

[JS92] C. S. Jensen and R. T. Snodgrass. Temporal specialization. In F. Golshani, editor, *Proceedings of the International Conference on Data Engineering*, pages 594–603, Tempe, AZ, February 1992. IEEE.

[LL97a] Mark Levene and George Loizou. The additivity problem for functional dependencies in incomplete relations. *Acta Informatica*, 34(2):135–149, 1997.

[LL97b] Mark Levene and George Loizou. Null inclusion dependencies in relational databases. *Information and Computation*, 136(2):67–108, 1 August 1997.

[LL98] M. Levene and G. Loizou. The additivity problem for data dependencies in incomplete relational databases. *Lecture Notes in Computer Science*, 1358:136–??, 1998.

[Mah94] Michael Maher. Constrained dependencies. In *Proceedings ILPS'94 Workshop on Constraints and Databases*, Ithaca, November 1994.

[Mah97] M. J. Maher. Constrained dependencies. *Theoretical Computer Science*, 173(1):113–149, February 1997.

[MS96] M. J. Maher and D. Srivastava. Chasing constrained tuple-generating dependencies. In *PODS '96. Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1996, Montréal, Canada, June 3–5, 1996*, volume 15, pages 127–138, New York, NY 10036, USA, 1996. ACM Press.

[Sal97] Eric Salvat. *Raisonner avec des oprations de graphes : graphes conceptuels et règles d'inférence.* PhD thesis, Université Montpellier II, Montpellier, France, December 1997.

[Sci82] Edward Sciore. A complete axiomatization of full join dependencies. *Journal of the ACM*, 29(2):373–393, April 1982.

[SF00] Iztok Savnik and Peter A. Flach. Discovery of multivalued dependencies from relations. Technical Report report00135, Albert-Ludwigs-Universitaet Freiburg, Institut fuer Informatik, Marseille 6, 2000.

[SM96] Eric Salvat and Marie-Laure Mugnier. Sound and complete forward and backward chainings of graph rules. In *Proceedings of the Fourth International Conference on Conceptual Structures (ICCS-96)*, volume 1115 of *LNAI*, pages 248–262, Berlin, August19–22 1996. Springer.

[Sow84] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, Reading, Mass., 1984.

[SSW94] K. Sagonas, T. Swift, and D. S. Warren. XSB as a deductive database. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 23(2):512–512, June 1994.

[SU82] Fereidoon Sadri and Jeffrey D. Ullman. Template dependencies: A large class of dependencies in relational databases and its complete axiomatization. *Journal of the ACM*, 29(2):363–372, April 1982.

[TS86] Hisao Tamaki and Taisuke Sato. OLD resolution with tabulation. In Ehud Shapiro, editor, *Proceedings of the Third International Conference on Logic Programming*, Lecture Notes in Computer Science, pages 84–98, London, 1986. Springer-Verlag.

[Ull88] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems. Volume I: Classical Database Systems.* Computer Science Press, 1988.

[Var84] Moshe Y. Vardi. The implication and finite implication problems for typed template dependencies. *Journal of Computer and System Sciences*, 28(1):3–28, February 1984.

[Vie86] L. Vielle. Recursive axioms in deductive databases: The query-subquery approach. In *1st Conference on Expert Database Systems, Kerschberg (ed)*, April 1986.

Using an Ordered Key Attribute in Database Design

Wilfred Ng
Department of Computing
Hong Kong Polytechnic University
email: csshng@comp.polyu.edu.hk

Abstract. Recently, we have extended the relational data model to incorporate linear orderings into data domains [8], which we call the ordered relational model. We herein formally define Ordered Functional Dependencies (OFDs) and Ordered INclusion Dependencies (OINDs) for the extended model. We show that the conventional sound and complete axiom systems for FDs and INDs can be generalised to the cases of OFDs and OINDs. We investigate a subclass of ordered databases, called ordered object databases, which consists of a set of ordered relations having a distinguished key attribute and enables us to view tuples as linearly ordered objects. An ordered object database possesses two desirable properties concerning OFDs and OINDs, which are useful in ordered database design. First, there is no interaction between OFDs and OINDs. Second, the implication problem for a given set of OINDs, I, whose complexity is polynomial-time in the size of I.

1 Introduction

Functional Dependencies (FDs) and *INclusion Dependencies* (INDs) [7, 2, 3] are commonly recognised as the most fundamental data dependencies that arise in practice in conventional relational databases. In the process of database design a desirable goal is to transform a database schema to be in *Boyce-Codd normal form* [6, 3] where FDs are a vital means to achieve the goal. However, it is not so common to incorporate INDs into the database design process as is the case of FDs. There are two main reasons. First, the implication problem for INDs is PSPACE-complete (c.f. see [2]), which is hard in complexity. Second, there is complicated interaction between FDs and INDs; in general case the implication problem of a mixed set of FDs and INDs is undecidable. The interaction can considerably slow down the algorithms needed in database design. A usual strategy is to study some very limited cases for INDs such as unary INDs in order to alleviate the two mentioned problems (see Chapter 10 in [6] for details).

Herein we assume that the data domains of the relational data model are linearly ordered and call the extended model the ordered relational model. Our extension is well-justified, since linear ordering is essential to the existing primitive data types used in DBMSs. There is also strong evidence that ordering is inherent to the underlying structure of data in many database applications [1, 8], in which linear ordering is particularly important to those advanced applications involving temporal information and scientific information [5, 4].

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 193–202, 2000.

We formalise the notions of FDs and INDs being satisfied in an ordered database and call them *Ordered Functional Dependencies* (OFDs) and *Ordered INclusion Dependencies* (OINDs). Informally speaking, OFDs can capture monotonicity between two sets of values and OINDs can capture the notion of the well-known *Hoare* power domain ordering [1]. The Hoare ordering can be viewed as a generalised subset ordering, representing the fact that one relation contains more information than another.

Consider in Figure 1 that a database consisting of two relations, called ALL_STAFF over the set of attributes {EMP, SALARY, POST, YEARS} and SALARY_BOUND over the set of attributes {MIN_SAL, MAX_SAL}. The semantics of ALL_STAFF and SALARY_BOUND are as follows: an EMPloyee with a given POST title, who has been working in a company for some YEARS, has the present SALARY. In addition, according to the company policy there are MINimum and MAXimum SALaries pre-defined for a staff member.

EMP	SALARY	POST	YEARS
Mark	40K	Senior Programmer	18
Wilfred	35K	Senior Programmer	15
Nadav	25K	Junior Programmer	7
Ethan	22K	Junior Programmer	6

(a) ALL_STAFF

MIN_SAL	MAX_SAL
20K	50K

(b) SALARY_BOUND

Fig. 1. Relations ALL_STAFF and SALARY_BOUND

We assume that there is a semantic ordering, 'Junior Programmer' $<$ 'Senior Programmer' in the domain over POST. The relation ALL_STAFF in Figure 1 then satisfies the OFD, {POST, YEARS} $\hookrightarrow$ SALARY, which states the fact that the SALARY of an employee is greater than that of other employees who have junior post titles and less experience in the company. Furthermore, if we assume that the salaries of all employees in the company are bounded by the minimum and maximum salaries, then this semantics can be captured by the following two OINDs, SALARY_BOUND[MIN_SAL] $\sqsubseteq$ ALL_STAFF[SALARY] and ALL_STAFF[SALARY] $\sqsubseteq$ SALARY_BOUND[MAX_SAL].

The implication problem for a mixed set of OFDs and OINDs is the problem of deciding for a given set Σ of OFDs and INDs whether Σ logically implies σ, where σ is an OFD or an OIND. We investigate the interaction between OFDs and OINDs. Our results show that OFDs and OINDs do not interact with respect to the collection and pullback rules, which are sound for the conventional FDs and INDs. On the other hand, we show that OFDs and OINDs interact by exhibiting a new mixed inference rule.

In order to investigate the usefulness of incorporating the OFDs and OINDs into database design in practice, we then restrict the scope of our investigation to ordered object databases, which are a special case of ordered databases that are constrained by having a single key attribute in each relation. An ordered object

relation enables us to view tuples as a set of linearly ordered objects, which is related to the recent research of capturing the data semantics of temporal complex objects [10]. We show that in the context of ordered object databases, OFDs and OINDs do not interact, and that the implication problems for OFDs and OINDs in this context are both decidable in polynomial-time. We emphasise that in our approach we have neither limited OINDs to be unary nor imposed the criteria of BCNF and key-based OINDs, which are common restrictions in conventional database design in order to achieve non-interaction between OFDs and OINDs (c.f. [6]). Thus, our result show that within the class of ordered databases, a good design principle in the presence of OFDs and OINDs is to impose an ordered key attribute for ordered relations.

Definition 1. (Notation) We refer to a sequence of attributes as a short hand for a sequence of distinct attributes and use the common notation for both sequences and sets. The fact that two sequences have the same elements is denoted by $X \sim Y$. The difference between two sequences of attributes, denoted as $X - Y$, is defined by the sequence resulting from removing all the common attributes in X and Y from X while maintaining the original order of the remaining attributes in X. We also denote by XY the *concatenation* of two sequences X and Y if X and Y are *disjoint*, or $X(Y - X)$ if X and Y are not disjoint.

2 Ordered Relational Databases

We assume the usual definition of a *linear ordering* $\leq$ on a set S, denoted by the structure $\langle S, \leq \rangle$. From now on, the term *ordered* will mean linearly ordered, unless explicitly stated otherwise. We assume that the equality predicate, $=$, still applies to ordered sets and use the notation, $<$, to represent the usual meaning of $\leq$ but $\neq$. We now define the concept of *pointwise-ordering* on the Cartesian product of ordered sets.

Definition 2. (Pointwise-Ordering) Let $D_1, \ldots, D_n$ be n ordered sets, t be an element in the Cartesian product $S = D_1 \times \cdots \times D_n$ and $t[i]$ be the ith coordinate of t. A *pointwise-ordering* on S is defined as follows: for all $t_1, t_2 \in S$, $t_1 \leq_S^p t_2$, if, for all $1 \leq i \leq n$, $t_1[i] \leq_{D_i} t_2[i]$.

Let D be a countably infinite set of constant values and $\leq_D$ be an ordering on D; without loss of generality, we assume that all attributes share the same domain D. We now give the definition of an ordered database.

Definition 3. (Attribute and Ordered Domain) We assume a countably infinite ordered set of attribute names, $\langle U, \leq_U \rangle$. For all attributes $A \in U$, the domain of A is $\langle D, \leq_D \rangle$. We call $\leq_D$ the *domain ordering* of D.

Definition 4. (Relation Schema and Database Schema) A relation schema (or simply a schema) R, is a subset of U consisting of a finite set of attributes $\{A_1, \ldots, A_m\}$ for some $m \geq 0$. A *database schema* is a finite set $\mathbf{R} = \{R_1, \ldots, R_n\}$ of relation schemas, for some $n \geq 1$.

Definition 5. (Projection) Let $X = \{A_1, \ldots, A_m\}$ be a finite subset of U where $A_i \neq A_j$ for $i \neq j$ and $A_1 \leq_U \cdots \leq_U A_m$. A *tuple* t over X is a member of D^m. We let $t[A_i]$ denote the ith coordinate of t. The *projection* of a tuple t onto a set of attributes $Y = \{A_{i_1}, \ldots, A_{i_k}\}$, where $1 \leq i_1 < \cdots < i_k \leq m$, is the tuple $t[Y] = \langle t[A_{i_1}], \ldots, t[A_{i_k}] \rangle$.

Definition 6. (Ordered Relation and Ordered Database) An *ordered relation* (or simply a relation) r defined over a schema R is a finite set of tuples over R. An *ordered database* (or simply a database) over $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a finite set $d = \{r_1, \ldots, r_n\}$ such that each r_i is a relation over R_i.

In our model we allow users to declare one or more semantic orderings relative to certain applications, which override the standard domain orderings in a DBMS. This approach is found to be useful in many applications, such as those involving temporal information. Readers may refer to [8] for more detailed discussion of various notions of orderings in our model and their applications.

3 Ordered Functional and Inclusion Dependencies

We now give the definition of an OFD. For the sake of simplicity in notation, we use $\leq_X^p$ to mean the pointwise-ordering on the Cartesian products of data domains associated with a sequence of attributes X in our further discussion.

Definition 7. (Ordered Functional Dependency) An *ordered functional dependency* (or simply an OFD) over a relation schema R, is a statement of the form $R : X \hookrightarrow Y$ (or simply $X \hookrightarrow Y$ whenever R is understood from the context), where $X, Y \subseteq R$ are sequences of attributes. The OFD $X \hookrightarrow Y$ is said to be *standard* if $X \neq \emptyset$. An OFD is satisfied in a relation r over R, denoted by $r \models X \hookrightarrow Y$, if, for all $t_1, t_2 \in r$, $t_1[X] \leq_X^p t_2[X]$ implies that $t_1[Y] \leq_Y^p t_2[Y]$.

We next give a set of inference rules for OFDs and show that Armstrong's axiom system carries over to ordered relations with respect to OFDs. We employ the usual notation $\mathrm{F} \vdash f$ to mean that f is *derivable* from F by a specified axiom system.

Definition 8. (Inference Rules for OFDs) Let X, Y, Z and W be subsets of R and F be a set of OFDs over R. The inference rules for OFDs are defined as follows:

(OFD1) *Reflexivity*: if $Y \subseteq X$, then $\mathrm{F} \vdash X \hookrightarrow Y$.
(OFD2) *Augmentation*: if $\mathrm{F} \vdash X \hookrightarrow Y$, then $\mathrm{F} \vdash XZ \hookrightarrow YZ$.
(OFD3) *Transitivity*: if $\mathrm{F} \vdash X \hookrightarrow Y$ and $\mathrm{F} \vdash Y \hookrightarrow Z$, then $\mathrm{F} \vdash X \hookrightarrow Z$.
(OFD4) *Permutation*: if $\mathrm{F} \vdash X \hookrightarrow Y$, $W \sim X$ and $Z \sim Y$, then $\mathrm{F} \vdash W \hookrightarrow Z$.

We remark that OFD4 is needed because we are dealing with sequences of attributes rather than the usual sets of attributes in FDs. The closure of a set of attributes X^+ in the context of OFDs F is given by $X^+ = \{A \mid \mathrm{F} \vdash \mathrm{X} \hookrightarrow A\}$. We now state the following theorem that the above axiom system is sound and complete for OFDs, holding in ordered databases. The underlying idea in this proof is standard [6, 3], where we also need to assume that each domain has at least two distinct elements (e.g. $\{0 < 1\}$).

Theorem 1. The axiom system comprising OFD1 to OFD4 is sound and complete for OFDs. □

We now give the definition of INDs and then review the inference rules constituting an axiom system, which was shown to be sound and complete for INDs by Casanova, Fagin and Papadimitriou [2]. For the sake of simplicity, we call this axiom system *Casanova et al.'s axiom system*. This result will be useful in proving our OINDs axiom system.

Definition 9. (Inclusion Dependency) An *inclusion dependency* (or simply an IND) over a database schema $\mathbf{R}$ is a statement of the form $R[X] \subseteq S[Y]$, where $R, S \in \mathbf{R}$ and $X \subseteq R$, $Y \subseteq S$ are sequences of distinct attributes such that $\mid X \mid = \mid Y \mid$. An IND is said to be trivial, if it is of the form $R[X] \subseteq R[X]$. An IND $R[X] \subseteq S[Y]$ over $\mathbf{R}$ is satisfied in d, denoted by $d \models R[X] \subseteq S[Y]$, whenever $\pi_X(r) \subseteq \pi_Y(s)$, where $r, s \in d$ are the relations over R and S, respectively.

We remark that INDs can also be defined over ordered databases as we have assumed the equality predicate over ordered sets.

Definition 10. (Casanova et al.'s Axiom System) Let I be a set of INDs over $\mathbf{R}$ and $R, R_1, R_2, R_3 \in \mathbf{R}$. *Casanova et al.'s axiom system* constitutes the following inference rules for INDs.

(IND1) *Reflexivity*: if $X \subseteq R$, then $\mathrm{I} \vdash R[X] \subseteq R[X]$.
(IND2) *Projection and Permutation*: if $\mathrm{I} \vdash R_2[Y] \subseteq R_1[X]$, where $X = \langle A_1, \ldots, A_m \rangle \subseteq R_1$, $Y = \langle B_1, \ldots, B_m \rangle \subseteq R_2$ and $i_1, \ldots, i_k$ is a sequence of distinct integers chosen from $\{1, \ldots, m\}$, then $\mathrm{I} \vdash R_2[B_{i_1}, \ldots, B_{i_k}] \subseteq R_1[A_{i_1}, \ldots, A_{i_k}]$.
(IND3) *Transitivity*: if $\mathrm{I} \vdash R_3[Z] \subseteq R_2[Y]$ and $\mathrm{I} \vdash R_2[Y] \subseteq R_1[X]$, then $\mathrm{I} \vdash R_3[Z] \subseteq R_1[X]$.

An OIND in the ordered relational model can capture the notion of Hoare orderings between two sets of values projected onto some attributes of two relations in a database, which arise naturally in those applications that consist of incomplete information [3]. The semantics of an OIND with two or more attributes on each side are also defined according to the pointwise-ordering extension on the Cartesian products of the underlying domains of the attributes in the OIND.

We now give the definition of an OIND as follows.

Definition 11. (Ordered Inclusion Dependency) An *ordered inclusion dependency* (or simply an OIND) over a database schema $\mathbf{R}$, is a statement of the form $R[X] \sqsubseteq S[Y]$, where $R, S \in \mathbf{R}$ and $X \subseteq R$, $Y \subseteq S$ are sequences of attributes such that $\mid X \mid = \mid Y \mid$. An OIND is satisfied in an ordered database d over $\mathbf{R}$, denoted by $d \models R[X] \sqsubseteq S[Y]$, if, $\forall t_1 \in r$, $\exists t_2 \in s$ such that $t_1[X] \leq_X^p t_2[Y]$, where $r \in d$ is the relation over $R \in \mathbf{R}$, and $s \in d$ is the relation over $S \in \mathbf{R}$.

From now on we will assume that when $R[X] \sqsubseteq S[Y] \in \mathrm{I}$ and d is a database over $\mathbf{R}$, then $r \in d$ is the relation over $R \in \mathbf{R}$ and $s \in d$ is the relation over $S \in \mathbf{R}$. The following proposition gives a simple relationship between OINDs and INDs.

Proposition 1. If $d \models R[X] \subseteq S[Y]$, then $d \models R[X] \sqsubseteq S[Y]$. □

We note that the converse of the above proposition does not hold as shown in the following simple counter-example.

Example 1. Let $d = \{r, s\}$ be a database over $\mathbf{R} = \{R, S\}$ where $r = \{0\}$ (1 tuple) and $s = \{1\}$ (1 tuple) are the relations over R and S, respectively, with $R = \{A\}$ and $S = \{B\}$. Then $d \models R[A] \sqsubseteq S[B]$ but $d \not\models R[A] \subseteq S[B]$.

We note that Casanova et al.'s axiom system [2] also applies to OINDs simply by replacing the occurrences of the symbol $\subseteq$ in an IND by the symbol $\sqsubseteq$ in an OIND. For the sake of uniformity in notation, we now re-label the three rules in Casanova et al.'s axiom system as OIND1, OIND2 and OIND3 when applying to OINDs in our further discussion. Casanova et al.'s axiom system is readily shown to be sound and complete for OINDs. The underlying idea of the proof uses the concept of a chase corresponding to a set of OINDs in order to formulate a counter-example of a database that satisfies I but not α (c.f. Theorem 3.1 in [2] and Theorem 10.9 in [6]). Note that Proposition 1 is utilised in proving the completeness of the following theorem.

Theorem 2. The axiom system comprising OIND1 to OIND3 is sound and complete for OINDs. □

Mitchell [7] presented a sound and complete axiom system for a mixed set of FDs and INDs, which contains two interaction rules between FDs and INDs [7] as follows, (1) the *pullback rule*, which derives a new FD from an FD and an IND, and (2) the *collection rule*, which derives a new IND from two INDs and an FD.

Definition 12. (Pullback Rule and Collection Rule for FDs and INDs) Let Σ be a set of INDs and FDs.

(Pullback): if $\Sigma \vdash R[WZ] \subseteq S[XY]$, with $\mid X \mid = \mid W \mid$, and $\Sigma \vdash S : X \rightarrow Y$, then $\Sigma \vdash R : W \rightarrow Z$.

(Collection): if $\Sigma \vdash R[UV] \subseteq S[XY]$, $R[UW] \subseteq S[XZ]$ and $\Sigma \vdash S : X \rightarrow Y$, then $R[UVW] \subseteq S[XYZ]$.

We now show that the pullback and collection rules are unsound for OFDs and OINDs.

Lemma 1. The pullback and collection rules are unsound for OFDs and OINDs in ordered databases.

Proof.
We use the following counter-examples to prove our claim.

Consider the database d_1 given in Figure 2. Let $\Sigma = \{R[AB] \sqsubseteq S[CD], S : C \hookrightarrow D\}$. Then it can be checked that $d \models \Sigma$ but $d_1 \not\models R : A \hookrightarrow B$. Consider another database d_2 given in the same figure. Let $\Sigma = \{R[AB] \sqsubseteq S[DE], R[AC] \sqsubseteq S[DF], S : D \hookrightarrow E\}$. Then it can be checked that $d \models \Sigma$ but $d_2 \not\models R[ABC] \sqsubseteq S[DEF]$. □

$d_1 = \{$ $r =$

A	B
0	0
0	1

$s =$

C	D
1	1

$\}$ $d_2 = \{$ $r =$

A	B	C
0	1	1

$s =$

D	E	F
1	1	0
0	0	1

$\}$

Fig. 2. Databases d_1 and d_2 show that the pullback and collection rules are unsound for OFDs and OINDs.

We observe that similar to the conventional case of FDs and unary INDs, there is no interaction between OFDs and unary OINDs in ordered databases. However, the interaction is rather complex in the general case. We now present one such interaction rule as follows.

Definition 13. (Interaction Rule for OFDs and OINDs) Let Σ be a set of OINDs and OFDs. If $\Sigma \vdash R[X] \sqsubseteq S[Z]$, $\Sigma \vdash R[Y] \sqsubseteq S[W]$, and $\Sigma \vdash S : A \hookrightarrow ZW$, with $A \in S$, then $R[XY] \sqsubseteq S[ZW]$.

It is straightforward to show that the above inference rule in Definition 13 is sound. We illustrate this inference rules by the following example.

Example 2. A special case of the rule is as follows, if $\Sigma \vdash R[A] \sqsubseteq S[B]$, $\Sigma \vdash R[C] \sqsubseteq S[D]$, and $\Sigma \vdash S : B \hookrightarrow D$, then $R[AC] \sqsubseteq S[BD]$.

We observe that the informal reason for the existence of the interaction between OFDs and OINDs is due to the fact that in an ordered relation there are many possible orderings arising from the projection on different combinations of attributes. In order to avoid the interaction, we need an attribute to co-ordinate all these orderings in an ordered relation. This idea will be further developed, resulting in our notion of ordered object relations discussed in the next section. We also remark that due to this reason we are not able to further strengthen the interaction rule in Definition 13 by generalising the singleton $A \in R$ into $P \subseteq R$ in the antecedent.

4 Interaction between OFDs and OINDs

We now define a subclass of ordered relations called *ordered object relations*. An ordered object relation formalises the notion of object identification in ordered relations. For instance, several DBMSs use a hidden attribute, containing a tuple identifier to perform sorting tuples. As an example consider the relation given in Figure 3. By using the ordered key YEAR to timestamp the tuples, we are able to capture the semantics of annual SALARY increments.

YEAR	SALARY
1997	15K
1998	18K
1999	20K

Fig. 3. A temporal relation using YEAR as an ordered key

Definition 14. (Ordered Object Database) Let $M \in R$ be a distinguished attribute, called an *ordered key attribute*. An ordered relation r over R is an *ordered object relation*, if $r \models M \hookrightarrow R$ holds. An ordered database is an *ordered object database*, if, for all $r \in d$, r is an ordered object relation.

We now extend our axiom systems for OFDs and OINDs to the class of ordered object relations by adding the following two inference rules.

(OFD4) *Ordered Key:* $\mathrm{F} \vdash M \hookrightarrow R$.

(OIND4) *Union:* if $\mathrm{I} \vdash R[A_i] \sqsubseteq S[B_i]$ for all $i \in \{1, \ldots, n\}$, then $R[X] \sqsubseteq S[Y]$, where $X = \langle A_1, \ldots, A_n \rangle$ and $Y = \langle B_1, \ldots, B_n \rangle$.

We remark that the above inference rules are crucial for preventing the interaction of OFDs and OINDs in ordered object databases. They imply that an OIND can be decomposed into an equivalent set of unary OINDs, leading to the non-interaction between OFDs and OINDs. It is interesting to note that OIND4 can be derived by using OFD4 and the inference rule in Definition 13.

Lemma 2. The inference rules OFD4 and OIND4 are sound for OFDs and OINDs in ordered object databases. □

A set of data dependencies Σ of OFDs and OINDs *logically implies* a data dependency σ over $\mathbf{R}$, written $\Sigma \models \sigma$, whenever for d over $\mathbf{R}$, if, for all $\sigma' \in \Sigma$, $d \models \sigma'$ holds, then $d \models \sigma$ also holds.

Definition 15. (Interaction between OFDs and OINDs) A set of OFDs F over $\mathbf{R}$ is said *not to interact* with a set of OINDs I over $\mathbf{R}$, if

1. for all OFDs f over $\mathbf{R}$, $\mathrm{F} \cup \mathrm{I} \models f$ if and only if $\mathrm{F} \models f$, and
2. for all OINDs α over $\mathbf{R}$, $\mathrm{F} \cup \mathrm{I} \models \alpha$ if and only if $\mathrm{I} \models \alpha$.

We now present our main theorem concerning the axiom system for a mixed set of OFDs and OINDs.

Theorem 3. The axiom system comprising OFD1 to OFD4 and OIND1 to OIND4 is sound and complete for OFDs and OINDs, holding in the class of ordered object databases.

Proof (Outline). The completeness of the system can be established by considering two cases. First, let $f = R : X \hookrightarrow Y$ be a non-trivial OFD such that $\mathrm{F} \cup \mathrm{I} \not\vdash f$. We need to show that $\mathrm{F} \cup \mathrm{I} \not\models f$. We construct an ordered object database d as given in Figure 4 to show that $d \models \mathrm{F} \cup \mathrm{I}$ but $d \not\models f$.

$d = \{r$ over R :

M	X^+	Z
1	$1 \cdots 1$	$1 \cdots 1$
0	$1 \cdots 1$	$0 \cdots 0$

Other relations r_i over $R_i \in \mathbf{R}$:

R_i
$1 \cdots 1$

$\}$

Fig. 4. An ordered object database d showing that $d \not\models R : X \hookrightarrow Y$

Second, let $\alpha = R[X] \sqsubseteq S[Y]$, where $X = \langle A_1, \ldots, A_n \rangle$ and $Y = \langle B_1, \ldots, B_n \rangle$, be a non-trivial OIND such that $\mathrm{F} \cup \mathrm{I} \not\vdash \alpha$. We show that $\mathrm{F} \cup \mathrm{I} \not\models \alpha$. We perform *the chase procedure* over the initial database d_0 given in Figure 5 with respect to I defined as follows: for any given OIND $R_1[T] \sqsubseteq R_2[W] \in \mathrm{I}$, if there exists $C_i \in T$ and $D_i \in W$ such that $\pi_{C_i}(r_1) = 1$ but $\pi_{D_i}(r_2) = 0$ (i.e. a violation is detected with respect to the OIND), then change the value for D_i in r_2 from 0 to 1. The resulting relation, d, obtained from the chase procedure, satisfies I. The proof can be concluded by showing that $d \not\models R[X] \sqsubseteq S[Y]$ for this case. □

$d_0 = \{r$ over R : $\begin{array}{|c|c|}\hline A_1 & R - A_1 \\ \hline 1 & 0 \cdots 0 \\ \hline \end{array}$ Other relations r_i over $R_i \in \mathbf{R}$: $\begin{array}{|c|}\hline R_i \\ \hline 0 \cdots 0 \\ \hline \end{array}\}$

Fig. 5. The initial ordered object database d_0 to be chased in order to show that $d \not\models R[X] \sqsubseteq S[Y]$

We remark that in the above proof we have not excluded non-standard OFDs. In other words, the theorem is valid for both standard and non-standard OFDs. The corollary below follows from the fact that in proving Theorem 3 we do not need to apply any inference rules involving a mixed set of OFDs and OINDs.

Corollary 1. OFDs and OINDs do not interact in ordered object databases. □

Corollary 1 implies that in ordered object databases the implication problem for a mixed set of OFDs and OINDs reduces to the two separate implication problems for OFDs and OINDs. These implication problems are easily solvable using the techniques in [6, 3]. We now give the formal definition of the *decomposition operator* $DEC(\mathrm{I})$, which allows us to show that in the context of ordered object databases the implication problem of OINDs can be solved in polynomial-time.

Definition 16. (Decomposition Operator) The *decomposition operator*, denoted as DEC, over an OIND $\alpha = R[X] \sqsubseteq S[Y]$, is given by $DEC(\alpha) = \{R[A_i] \sqsubseteq S[B_i] \mid A_i \in X \text{ and } B_i \in Y\}$, with $X = \langle A_1, \ldots, A_n \rangle$ and $Y = \langle B_1, \ldots, B_n \rangle$. The decomposition of a set of OINDs, I, denoted by $DEC(\mathrm{I})$, is defined by $DEC(\mathrm{I}) = \bigcup_{\alpha \in I} DEC(\alpha)$.

Using the decomposition operator we are able to reduce the implication problem for OINDs, into the simpler implication problem for unary OINDs.

Lemma 3. Let $X = \langle A_1, \ldots, A_n \rangle$ and $Y = \langle B_1, \ldots, B_n \rangle$. $\mathrm{I} \vdash R[X] \sqsubseteq S[Y]$ if and only if $DEC(\mathrm{I}) \vdash R[A_i] \sqsubseteq S[B_i]$ $\forall i \in \{1, \ldots, n\}$. □

We now formally state our result concerning the implication problem for a mixed set of OFDs, F, and OINDs, I. The next theorem follows immediately from Corollary 1 and Lemma 3.

Theorem 4. The implication problem for a mixed set of OFDs, F, and OINDs, I, in ordered object databases can be solved in polynomial-time in the sizes of F and I. □

As a consequence of Theorem 4, we suggest that a good database design incorporating OFDs and OINDs can be achieved in the context of ordered object database. By considering OFDs and OINDs in a database design process, a database designer can capture the ordering semantics of the database schema in an easy manner. It is important to note that the only assumption that we made is the existence of an ordered key for an ordered relation, which is a simple and natural constraint embedded in many database applications, as evidenced

by the research for many years [4, 5, 9]. Apart from this assumption, we have not imposed any restriction on the form of INDs as is the case of standard databases [6].

5 Conclusions

We have defined ordered databases and introduced OFDs and OINDs. We have studied the implication problems of OFDs and OINDs, and presented their respective sound and complete axiom systems in Theorems 1 and 2. In order to study whether mixing OFDs and OINDs is easy to handle in database design, we have investigated the interaction between OFDs and OINDs and found that the known interaction rules are unsound. By restricting our scope to an important class of ordered relations having a single ordered key attribute we formalised ordered object databases in Definition 14, allowing us to view an ordered relation as a set of ordered objects. In the context of ordered object databases, we have shown in Theorem 3 a good result that OFDs and OINDs do not interact. By using the fact that the implication problem for OFDs and unary OINDs is polynomial-time decidable, we defined a decomposition operator to transform such OINDs into a corresponding set of unary OINDs, concluding in Theorem 4 a surprising result that the implication problem for a mixed set of OFDs and OINDs is also polynomial-time decidable. Our results suggest that a good database design strategy for linearly ordered objects in the presence of OFDs and OINDs is to restrict a relation to have an ordered key attribute, in this case OFDs and OINDs have no interactions and thus incorporating a mixed set of OFDs and OINDs in database design is practically feasible.

References

1. P. Buneman et al. Using Powerdomains to Generalise Relational Databases. *Theoretical Computer Science* **91**, pp. 23-55, (1991).
2. M.A. Casanova et al. Inclusion Dependencies and their Interaction with Functional Dependencies. *Journal of Computer and System Science* **28**, pp. 29-59, (1984).
3. M. Levene and G. Loizou. *A Guided Tour on Relational Databases and Beyond.* Springer-Verlag, London, (1999).
4. N.A. Lorentzos. DBMS Support for Time and Totally Ordered Compound Data Types. *Information Systems* **17**(5), pp. 347-358, (1992).
5. D. Maier and B. Vance. A Call to Order, In *ACM symposium on Principles of Databases Systems*, pp. 1-16, (1993).
6. H. Mannila and K-J Raiha. *The Design of Relational Databases.* Addison-Wesley, (1992).
7. J.C. Mitchell. The Implication Problem for Functional and Inclusion Dependencies. *Information and Control* **56**, pp. 154-173, (1983).
8. W. Ng and M. Levene. The Development of Ordered SQL Packages for Modelling Advanced Applications. In *LNCS 1308: DEXA '97, Proceedings,* Springer-Verlag, pp. 529-538, (1997).
9. P. Seshadri et al. The Design and Implementation of a Sequence Database System. *VLDB '96 Conference, Proceedings,* pp. 99-110, (1996).
10. J. Wijsen. Temporal FDs on Complex Objects. *ACM Transactions on Database Systems* **24**, pp. 227-176, (1999).

Awareness in Interactive Database Applications

Günther Specht[1], Patrick Freiherr Harsdorf von Enderndorf[2], Thomas Kahabka[3], and Franz Peterander[2]

[1] Institut für Informatik
Technische Universität München
D–80290 München, Germany

[2] Institut für Pädagogische Psychologie und Empirische Pädagogik
Ludwig Maximilian Universität München
D–80802 München, Germany

[3] Exolution GmbH
Barerstr. 60, D–80799 München, Germany

specht@in.tum.de, patrick@harsdorf.de, kahabka@exolution.de,
peterander@lrz.uni-muenchen.de

Abstract. This paper proposes a way to overcome locking problems in interactive database applications by using awareness concepts. Parallel long running editing sessions in interactive database applications often cause locking conflicts. The occurrence of conflicts can be drastically decreased by giving users means to be aware of each other and to communicate. This paper explains how this is done in database applications for the social area, which are prone to locking conflicts due to their use of relations with a very large number of attributes and long running transactions. Additionally it shows how high scalability can be achieved with the help of dynamic partitioning schemes. We present MAL, a development system for interactive database applications, which allows to develop applications that automatically include awareness-based locking in a network environment.

1 Introduction

In multiuser database applications used for long-running editing sessions severe locking problems are often encountered. The probability of these conflicts increases with the number of attributes per tuple and the duration of editing times.

We faced such problems in the development of applications for *early childhood intervention centers*, institutions providing help to children with development problems. Such database applications assisting in social diagnostics use relations containing up to 3000 attributes, all functionally dependent on the patient-identifier. Since editing sessions for diagnostic reports often take over one hour, locking conflicts are a problem.

We offer a new awareness-based approach to overcome these locking problems and have implemented it in the MAL system, a database application development

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 203–211, 2000.

system successfully used in the social area. MAL was developed by the MAL Team, an interdisciplinary research group.

The rest of the paper is organized as follows. In the next section we introduce the awareness-based locking approach of MAL. In section 3 we present techniques to maintain high scalability. Section 4 is devoted to a closer description of the MAL system. Conclusions are discussed in section 5.

2 Locking and Awareness

2.1 The Gallery

Instead of using strict isolation and anonymity between different users the concept of making users aware of each other gives them a chance to cooperate. The central element of the awareness interface is a floating window called "the gallery" (figure 1). It informs users about which of their co-workers are currently logged in, and which co-workers are using the same record as they do. It can also be used to communicate with others by exchanging text messages.

Each co-worker is identified by a unique color in the gallery which is used to draw the border of his or her image. This color is used to identify screen elements locked by different users working on the same record.

Fig. 1. The "Gallery"

Making users aware of the virtual presence of their co-workers not only facilitates work but also creates a feeling of community. The MAL system uses pictures instead of names – which would consume less screen real estate – , because pictures are recognized faster than words and can be perceived even if not focused directly [7]. Also in a cooperative multiuser environment subjective satisfaction, inclination to cooperate and efficiency increase as the users establish a mental connection between the virtual representation of their co-workers and the actual persons [1]. For that effect, pictures – being more personal and direct – are more appropriate [6].

2.2 Locking and Awareness

In MAL applications locking has multiple functions: It is not just seen as a necessary evil to gain consistency and prevent data loss. Rather, the visual representation of locks represents awareness information about the presence of co-workers.

While classical database applications are based on ACID transactions using strict isolation and anonymity between different users, MAL's awareness concept surrenders the anonymity of locks. Whenever a user encounters a locked attribute its value is visible along with the information about who is working on that attribute. In addition we surrender the concept of isolation. If a user stays on a user interface (UI) page with locked attributes, he can see the values change with a certain delay as his team-mate modifies them. Thus we allow informed dirty reads. The person currently working on that data can recognize the virtual presence of his colleague, who just "stepped into the room". The gallery is extended and the fields in the form are colored correspondingly. Two MAL users working on the same database record are very close to what is a shared desktop in *relaxed WYSIWIS (what you see is what i see)* mode [4]. The MAL concept takes advantage of the fact that people tend to transfer behaviour patterns and emotions concerning physical rooms and spaces over to virtual spaces. This encourages a form of natural and intuitive cooperation, comparable to the one existing between athletes sharing training machines. An athlete who is just about to finish his exercises or is just resting will step back as soon as possible to free the machine for an arriving colleague. On the other hand, if he is still using the machine and the other one stays waiting, the former will give some informal information like "just a moment" or "give me ten more minutes".

Whenever two MAL users virtually meet in a MAL application, both have the option to initiate a conversation by clicking on the picture of the co-worker. In most occasions awareness of the presence of the other and maybe a personal communication via the gallery will solve a locking problem. A polite and cooperative conduct is encouraged through contextual awareness [2]. Awareness about co-workers helps to reduce frustration and increase the efficiency of cooperative work [3]: "Seeing" co-workers often prevents potential conflicts from occurring, since it is possible for a user to intuitively understand his co-worker's task and to correctly predict their next steps.

2.3 Intuitive User Interface supporting Awareness-based Locking

One of the primary goals in creating the MAL system was to keep the user interface of the MAL applications as intuitive as possible. An interface like that of many multiuser text editors, where a user has to claim temporary ownership over a chunk of data before he can modify it [5] would consume too much time and distract the users from their tasks. Instead we propose a straightforward and intuitive user interface which does not require additional buttons, commands and procedures to remember.

Therefore we introduce the new concept of wish locks, with the intended meaning to hold a (probably dirty) read lock and to wish (and to wait for) a

write lock. Wish locks are always immediately obtainable. When a user enters a UI page for editing, the MAL application automatically locks all unlocked attributes on this UI page with write locks and all locked variables with wish locks. In either case the page can be displayed immediately.

If user A enters a UI page containing attributes locked by user B, the MAL system behaves as follows. On user A's side the editing fields of the corresponding attributes are disabled and their labels are crossed out in user B's color. On user B's side the labels of the same editing fields are marked with a "W" ("wish") in user A's color, to inform user B that he is blocking those attributes for user A (figure 2). To keep the user interface simple there are no dedicated buttons or commands for releasing locked attributes. To do this, users "step away" from them, i.e. move to another UI page. Clients always release all wish- and write-locks whenever the user changes from one page to another, even if both pages use the same variables. Users waiting for a lock to be released form a queue (FIFO) and are guaranteed to be granted access in the correct order – independent of network lag. Likewise, if a user quickly moves one screen back and forth while another user is waiting for locks on the the first screen to be released, the access rights are guaranteed to be granted to the second user and not be reclaimed by the first one.

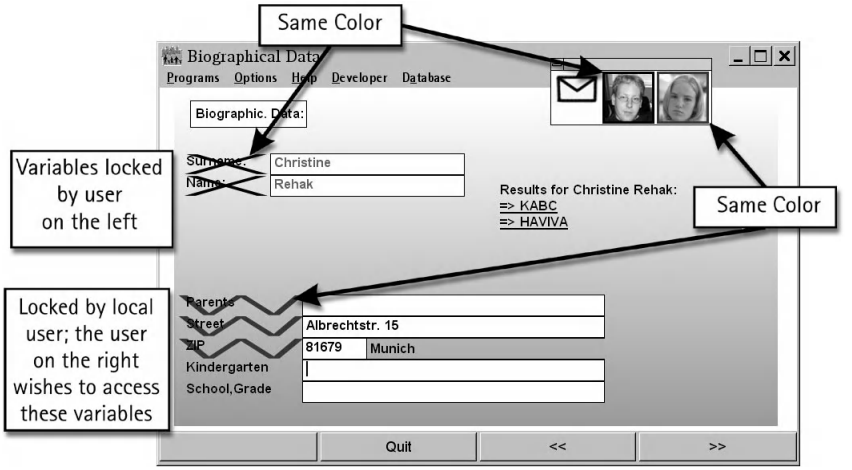


Fig. 2. User Interface

2.4 Implementation

For the implementation of our awareness-based locking concept MAL uses its own locking scheme on top of the locking mechanism of the underlying database. This is done with special database tables containing locking information and

messages sent directly from client to client. Notification mechanisms are tightly integrated into the MAL system. If a user changes the value of an attribute the change of this attribute will not only be transmitted to all other MAL clients working on the same record, but also all results that depend on this value will be recomputed. In this way displayed results – even diagrams – react in near real-time to changes made by other users.

The relations used by MAL applications often contain very large numbers of attributes. On the database server these relations get transformed to name-value pairs, mapping every attribute of the relation to a row in the database. This allows the minimum lockable unit to be a single attribute.

For optimal efficiency the database server should support row level locking. If a database supports only page level locking it may occur that MAL clients cannot get write access to attributes even though no other MAL client deliberately locked them, since they reside on the same database page as a locked attribute. Since locking information is held in MAL in separate lock tables, physical database locks are only held very shortly. In this case short delays can result.

MAL applications are by their nature deadlock free. Input elements in an MAL application are always tied to exactly one attribute. Cyclic deadlocks on different UI pages cannot occur, since all write- and wish-locks of one UI page are claimed in an atomic step (preclaiming) and completely released on each page change. Thus locks cannot be subsequently demanded. Lifelocks are avoided since wish locks form a FIFO queue.

3 Scalability

Since MAL-based applications may be used in larger institutions such as hospitals, the network database subsystem was built with scalability in mind. A large part of the awareness functionality and the propagation of changed values is done through direct client to client connections, thereby reducing load on the central server. Nevertheless the tables containing locking information and the tables containing the actual data reside on the database server and are shared by all clients. These tables may become heavily used and could become potential hotspots. Therefore these tables can be horizontally partitioned, reducing lock contention and increasing performance.

For the partitioning to have optimal benefit, the goal is not to have partitions of equal size, but to have the amount of insert and delete operations fairly equally distributed. Therefore the number of insert and delete operations is counted for each attribute.

The balancing mechanisms of the MAL system are as follows:

- New attributes are always created in the least-used partition.
- The first MAL client to connect to a given MAL database on a given day moves attributes from more heavily used partitions to lesser-used ones to balance them. This takes only fractions of a second and goes unnoticed by the user. Since MAL databases only change incrementally, the balance

between partitions does not deteriorate rapidly and thus balancing – akin to file system defragmentation – does not have to be performed often. As no data needs to be moved among partitions while the databases are in use, the clients can cache the information about which attributes resides in which partition locally. The initial number of partitions per table is freely configurable. The system administrator can instruct the MAL system to add more partitions to an existing table at any time. Data will then be moved automatically from the original partitions to the new ones until once again all partitions are balanced. On networks with several database servers, each partition can reside on a different database server, significantly increasing performance. Since real-time bookkeeping of the insert and delete operations could become a performance issue in itself, every MAL client logs its insert and delete operations and updates the information just once per session.

The effect of partitioning on performance (figure 3) is as follows: The performance at first increases in a near logarithmic fashion with each added partition then reaches a plateau. Finally as more and more partitions are added, a slow linear decrease in performance is measured, which can be attributed to administrative overhead.

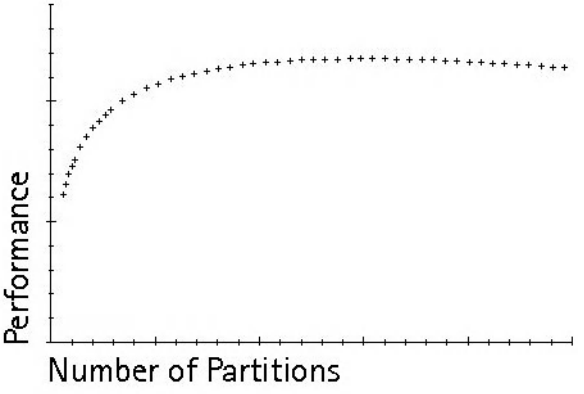


Fig. 3. Effect of partitioning on Performace

Since the effectiveness of partitioning depends on the number of occurring conflicts, the optimal number of partitions varies with the number of clients typically connecting, access patterns of the users, the underlying hardware, and the database management system.

Partitioning significantly reduces lock contention and allows a large number of MAL clients to share a common database and still maintain high response times.

4 Description of the MAL System

MAL was initially created to develop applications for *early childhood intervention centers*, and its major use is still in that area.

4.1 Early Childhood Intervention

Early childhood intervention is a service provided to children and their families in many countries. The goal is to detect potential problems in the development of children at an early stage. By administering special care to these children it is then possible to prevent these development problems from occurring, remedy existing problems or at least lessen their impact. In early childhood intervention centers, specialists of different medical and therapeutical professions are working together. MAL applications are mostly used to assist in diagnosis, and to pool and structure knowledge of different experts over several sessions and to help for example in the creation of reports. Additionally many standardized tests like KABC have been implemented as MAL applications.

4.2 Database Application Development using MAL

The MAL applications are created in a different setting than those in the classic commercial, administrative or technical fields. We decided to build a new software development system because of the different circumstances in the social area:

- It is not possible for the users to give in advance a clean definition of what a program that is developed for them shall do: The software development process is highly iterative with a lot of trial-and-error between the developers and the users. This leads to a high change frequency in the application's data scheme and program logic.
- The software developer has to understand the needs of the users. He has to be a specialist in the social area to be able to communicate about the wishes of the users: We can not assume the programmer to have classic software-design and development skills. The system has to give the developer as much freedom as he needs but also protect him from faults and inconsistencies that might be easily introduced because of the high change frequency.
- The programs are written for computer-illiterates who might even fear using a computer. The software that is developed with MAL has to be very easy to use and present a consistent user interface. The programs have to allow the user instant access to the data and let him interactively "play" and analyze the stored data. This requires a seamless but generic integration of the data analysis functions with the user interface elements.

To cope with these requirements we decided to create a rapid software development system based on a newly designed language for the definition of the data-processing structure, the form-sets and the form-transition-graph. The MAL

language shares many concepts and features with functional languages such as prevention of side effects, lazy evaluation, declarative structure.

To write a MAL program the author has to define the user interface and the data structure with the functional data processing structure. The definition of the user interface is done by designing each page by placing predefined user interface elements on a virtual grid on the screen. Each input element is bound to a attribute in the database, where the value entered by the user is almost immediately stored. Each output element is bound to either a database attribute or a MAL function, whose results it displays. A MAL function is defined by combining values of database attributes or other MAL functions with the help of predefined functions or operators. Database attributes and functions can store or respectively return only values of two types: numbers and texts. There is no way of assigning a value to a database attribute other than binding it to a user interface input element. That means, the value of all functions is at all times a direct function of the user-entered values: the internal state of the program is entirely bound to the values entered by the user. A change of state can only happen by the user altering a value in a user interface input element.

The user interface and the functional structure are internally tightly coupled and kept consistent. That means, if an output-element is on the same form as an input element, and the function that is bound to the output-element uses (directly or indirectly via multiple levels of functions) the database attribute bound to an input element on the same form, the output-element updates automatically whenever the user alters the value of the input element. The programmer only defines the function-structure behind it – the MAL system takes care of screen-updates.

5 Conclusions

In this paper we presented awareness concepts for avoiding locking conflicts in interactive database applications with long running transactions. While classical database applications are based on ACID transactions using strict isolation and anonymity between different users, our awareness concept surrenders isolation and the anonymity of locks. Making concurrent users aware of each other gives them a means to cooperate on a meta level about restricted resources like long-locked tuples. We introduced wish locks to express write wishes to the owner of write locks and to gain intermediate results. An intuitive user interface supporting awareness-based locking was introduced. Scalability can be reached by the use of vertical and dynamic horizontal partitioning. We implemented these concepts in the MAL system, a rapid database application development system. Several applications in the area of *early childhood intervention* using MAL are already in use.

References

1. Elena Rocco, University of Michigan: *Trust Breaks Down in Electronic Contexts but Can Be Repaired by Some Initial Face-to-Face Contact.* Conference on Human

Factors and Computing Systems (CHI), 1998, Los Angeles, pp. 496 – 502
2. Gloria Mark, Ludwig Fuchs, Markus Sohlenkamp: *Supporting Groupware Conventions through Contextual Awareness.* Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW), 1997, Lancaster, pp. 184 – 193
3. Carl Gutwin, University of Saskatchewan: *Effects of Awareness Support on Groupware Usability.* Human factors and Computing Systems (CHI), 1998, pp. 511 – 518
4. Johann Schlichter, Michael Koch, Chengmao Xu: *Awareness – The Common Link Between Groupware and Community Support Systems.* Lecture Notes in Computer Science 1519, Community Computing and Support Systems, T. Ishida (ed.), Springer Verlag, 1998, Berlin, pp. 77 – 93
5. Johann Schlichter, Michael Koch, Martin Bürger: *Workspace Awareness for Distributed Teams.* Proc. Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents, Singapore, Lecture Notes on Computer Science 1364, W. Conen, G. Neumann (eds.), Springer Verlag, 1997, Berlin, pp. 199 – 218
6. Carl Gutwin, Saul Greenberg, Mark Roseman: *Supporting Awareness of Others in Groupware.* Conference on Human Factors and Computing Systems (CHI), Vancouver, 1999, pp. 205 ff.
7. Paul Dourish, Victoria Bellotti: *Awareness and Coordination in Shared Workspaces.* Conference proceedings on Computer-supported cooperative work, 1992, Toronto, pp. 107 – 114

WorkMan - a Transactional Workflow Prototype

Juha Puustjärvi[1] and Harri Laine[2]

[1] Helsinki University of Technology, Laboratory of Information Processing Science,
P.O.Box 5400
FIN-02015 HUT, Finland
Juha.Puustjarvi@cs.hut.fi
[2] Department of Computer Science, P.O.Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
Harri.Laine@cs.helsinki.fi

Abstract. Workflows are activities involving the coordinated execution of multiple tasks performed by different processing entities. Workflow management systems support the specification and execution of workflows. WorkMan is a prototype of a workflow management system in which considerable attention is paid for the utilization of the services of database systems. Particularly the use of SQL-features in implementing workflow scheduling and supporting the transactional properties of workflows are investigated. In the WorkMan prototype the scheduling of tasks is based on SQL-triggers while the transactional properties of workflows (isolation and atomicity) are implemented using SQL-assertions and CHECK-constraints.

1 Introduction

The purpose of a workflow management system is to support the specification and execution of business processes. A workflow specification describes the tasks of a business process, their dependencies, and the requirements these tasks impose on information system functionality and on human skills. The combination of workflow systems and database management systems significantly facilitate the specification and reliable management of complex business processes.

Most workflow-related research done in the past few years can be categorized into workflow design and specification, inter-task dependency specification and management, and workflow management system design.

In this paper, we will restrict ourselves on workflow management system design. Particularly we will present how task scheduling and the management of workflows' transactional properties are implemented in the WorkMan (Workflow Manager) prototype. The focus in the WorkMan project has been on investigating different approaches of utilizing the services of the database management systems and the standards of distributed computing. Primarily the feasibility of SQL-triggers in implementing workflow executions, and SQL-assertions in enforcing workflows' transactional properties are investigated.

SQL-triggers are *event-condition-action* rules (*ECA rules for short*)[14]. The *event* part contains a list of database events which cause the rule to be triggered.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 212–221, 2000.

The *condition* part contains a boolean expression of predicates on a relation. The condition is checked when an event listed in the event part occurs. The *action* part contains a sequence of operations which are executed when the condition is true. The trigger statement in SQL3 gives the user a number of different options in the event, condition, and action parts [12].

Using active rules in scheduling workflow tasks is not a new idea. The use of active rules for managing workflows was proposed originally in [3]. Later on the use of active rules was proposed e.g., in [2] where also the automatic specification of active rules was proposed.

In WorkMan the workflow designer uses a CASE-tool for specifying the inter-task execution dependencies within a workflow [7]. The specification, that will be stored in XML, then acts as the source for generating the SQL-triggers and the internal data structures for task scheduling. Because the underlying database management system takes care of the triggers, there remains very little to do for a specific workflow engine in task scheduling.

In the WorkMan prototype the workflow failure atomicity is enforced either by compensating transactions or by *options* [10]. An option is a certification of the success of a potential later update. Instead of performing updates a workflow instance acquires options, and if the workflow instance can be committed then the options are eventually realized by installing the updates to the database. Through options we avoid the problems of dirty data.

Mere failure atomicity, is not a sufficient correctness criterion for workflows. Workflows should also behave correctly in the presence of other concurrent workflows. In WorkMan the notion of a *self-isolating workflow* [10] is used for workflow concurrency control. Such a workflow sets and unsets *restrictions* in a way similar to the way in which traditional transactions obtain and release locks.

The rest of the paper is organized as follows. Sections 2 overviews how failure atomicity can be enforced by options. Section 3 outlines the idea of self-isolating workflows. The notion of *state relations* is introduced in Section 4. The architecture of the WorkMan system is presented in Section 5. Through the architecture we outline how workflow specifications are compiled into triggers, and how the tasks are scheduled. In Section 6 we illustrate with an example our proposed notion and how they are interwoven. Section 7 concludes the paper.

2 Workflow failure atomicity

A workflow supports *semantic atomicity* [5], if immediately after the commitment of a subtransaction (e.g., a task) other transactions are allowed to access the updated data (dirty data) [1], though the workflow may still be aborted. Abortion is processed by executing compensating transactions for each subtransaction that has already committed. So each compensating transaction "undoes", according to the semantics of the application, the effects of the subtransaction being compensated. For example the transactional workflows based on flexible transactions [4], negotiation transactions [9] and Sagas [6] support semantic atomicity.

A problem with semantic atomicity, however, is that the access to dirty data may yield to inconsistent results, or may cause a subsequent compensating action to fail. On the other hand, if the effects of dirty data are avoided by cascading aborts, then the workflow management system has to keep track of all workflows which have accessed dirty data. In the case of workflow abortion, all the tasks which accessed dirty data and all their successors have to be aborted or compensated.

By setting an option a workflow instance ensures that a certain potential later update of the data item will not fail as a result of other concurrent activities. Options are implemented by two-phase tasks. A *two-phase task* (or a *2P-task* for short) is a decomposition of a task into two tasks: the first one, called the *preliminary task*, ensures that the second one, called the *complementary task*, is always executable, i.e., it will not fail as a result of other concurrent activities. Each complementary task comes in two versions: a *positive complementary task* and a *negative complementary task*. Only one of them is executed in a specific task instance.

3 Workflow isolation

Our idea is to use consistency constraints in the same way as locks are used in traditional concurrency control. We call consistency constraints as *data restrictions* when they are used in concurrency control. Data restrictions differ from traditional consistency constraints in that they are short-lasting.

Logically, data restrictions have similarities with *escrow locking* [8]. Escrow locking was originally designed to deal with arithmetic operations on hot spot data. Later on with the Contract model [11,13] escrow locking was proposed to be used in ensuring execution atomicity. Our approach differs from escrow locking in that instead of predicates we evaluate only consistency constraints (predicates) that can be expressed using SQL.

Although workflow isolation requirements can be enforced by data restrictions, in some cases it seems to be more appropriate to control the execution of workflows directly, i.e., restricting the execution of the programs behind the workflows rather than indirectly through the data they access. We call these restrictions *workflow restrictions*. They differ from traditional semaphores [1] in the same way as data restrictions differ from traditional locks, i.e., the locking expression is based on semantic information (expressed using SQL).

Both data and workflow restrictions are given in the workflow specification. They are then transformed into tasks. From the system point of view setting a restriction or unsetting a restriction is like any task of the workflow. A workflow instance then sets and unsets restrictions according to the specification. Due to this property we call such workflows *self-isolating workflows*.

4 State relations

Workflow systems store information about how the workflow instances proceed, e.g., which tasks are active, terminated or failed. Such *state information* is needed when the users are querying the state of certain workflow instances. WorkMan system uses the state information also in trigger based task scheduling as well as in workflow concurrency control.

We make some assumptions of the form of the state information. Particularly, we assume that for each workflow there is a *state relation*. State relations are named according to the names of the corresponding workflows. They have an attribute to identify the instances, attributes for each task of the workflow, and an attribute for each parameter appearing in the workflow.

An important point in using workflow restrictions and state relations in workflow concurrency control is that before a workflow instance, is allowed to start, it must succeed in inserting a tuple into the state relation. However, the insertion fails if there is a workflow restriction on the state relation that would evaluate to false after the insertion. Respectively there may be a workflow restriction which may prevent the modification of a state of a workflow instance.

5 WorkMan Architecture

The WorkMan incorporates the following functions:

- support for form based workflow specification,
- analysis techniques to verify the correctness of the specifications,
- trigger based task scheduling,
- workflow concurrency control based on self isolating workflows, and
- workflow recovery based on options or compensating transactions.

The architecture of the WorkMan is presented in Figure 1.

The *workflow specification tool* provides the interface for workflow designers. They specify the workflows by forms and diagrams. The specification is stored in XML-format. The *XML analyzer* analyzes the specification to determine whether such a workflow can be implemented. For example, it is ensured that the tasks of the workflow are specified. It also provides reports and overviews about the specifications.

The *trigger generator* translates the XML-specifications into SQL specifications of database tables and triggers. Let us assume that the workflow W has n tasks. Then the trigger generator generates for W

- the workflow state relation W
- a *start trigger*, for each task appearing in W, i.e., n triggers altogether,
- a task execution *request relation* for each task appearing in W,
- a task execution *response relation* for each task appearing in W, and
- a *return trigger*, for each response relation.

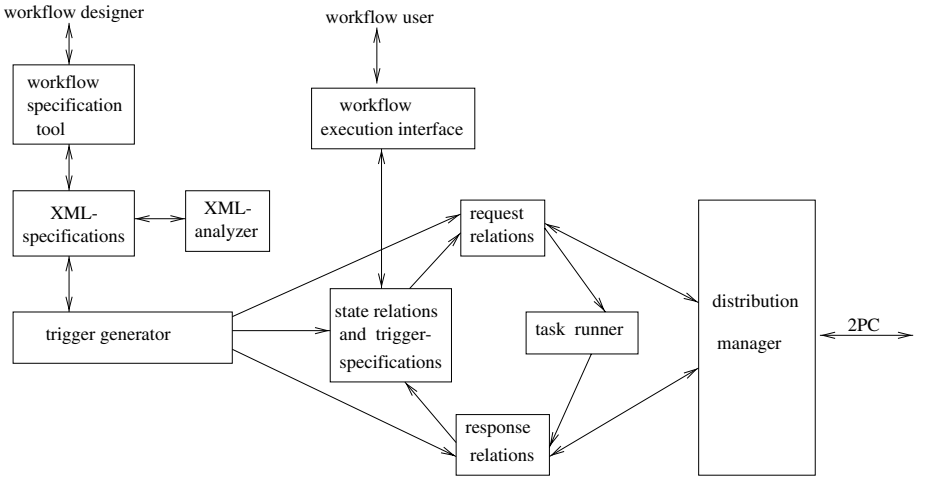


Fig. 1. WorkMan architecture

The state of each workflow instance is represented by a tuple in the workflow state relation, and the coordination of the execution of its tasks is enforced by the triggers placed on the state relation and the triggers placed on the response relations.

The execution of a task divides into three successive transactions. First, a start trigger generates an insertion (an execution request) into a task request relation. Second, the *task runner* executes the following actions:

- removes the execution request from the request relation,
- executes the request, and
- inserts a reply into a *response relation.*

Third, a return trigger removes the reply from the response relation and updates the workflow state relation accordingly.

Distributed execution of tasks matches nicely to this architecture: if an execution request is inserted at site i into a request relation but the task to be executed locates at site j, then the *distribution manager* transfers the execution request into the request relation of site j. This trasfer is executed under the control of the 2PC-protocol [1]. Respectively, after the execution of the task at site j, the response is transfered from site j to site i under the control of the 2PC protocol. Note that request relations, response relations and workflow state relations are in a stable storage. So the recovery system can decide the state of each task after a failure, and so also the atomicity of remotely executed tasks is ensured.

The *task runner* module executes the tasks by communicating with the entities that process the tasks. Typically the processing entity is a database system.

The *workflow interface* is a module which provides the interface between the user and the workflow system. A user activates a workflow instance by passing a workflow name and the necessary parameters to the workflow interface, which starts the instance by inserting a tuple into the workflow state relation. The inserted tuple contains only the identifier of the instance (key attribute) and the input parameters of the activated instance.

6 Workflow example

In this section we will illustrate by an example the notions of options, restrictions, and trigger base workflow scheduling. We will not present the forms that the workflow designer uses in specifying the workflow, neither, the XML-format of the specifications. For illustrative purposes we present an example workflow, called *Loan request* workflow, in a graphical form (Figure 2.)

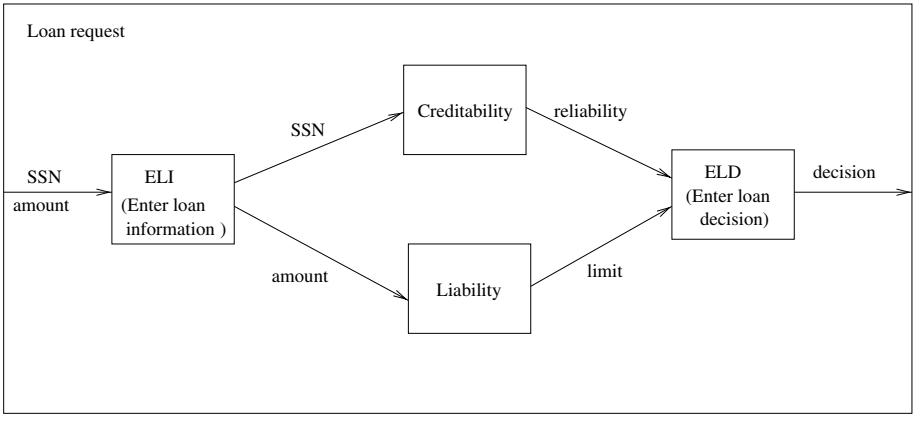


Fig. 2. Workflow Loan request

The first task *Enter loan information* accepts the requested loan amount and the SSN (social security number) of the client. Then based on the SSN of the client the task *Creditability* evaluates client's creditability and outputs the parameter *reliability* for the task *Enter loan decision*. The task *Liability* checks that the bank does not exceed its liability limit if the requested loan amount is granted. Its output parameter *limit* indicates the liability limit of the bank. The task *Enter loan decision* either grants or refuses the loan request. In addition, f the loan is granted, then the information of the grant is stored and bank's total liability is incremented by the amount of the loan.

6.1 Task scheduling in the example workflow

As stated in Section 4 in order to support the state of each workflow instance there is a state relation for each workflow. We name such relations according to the corresponding workflows. So, with respect to the workflow *Loan request* there is the state relation *LoanRequest* (Figure 3). It has

- as a key attribute the workflow instance identifier (denoted insId),
- one state attribute for each task appearing in the workflow (denoted by S followed by the name of the task), and
- one attribute for each input or output parameter presented in the workflow (denoted by the name of the parameter).

LoanRequst	insId	SSN	amount	S-ELI	S-Creditability	S-Liability	limit	S-ELD	decision
	0012	3232	500	C	—	—	—	—	—
	1003	5252	470	—	—	—	—	—	—
	1111	1234	1000	—	—	—	—	—	—

Fig. 3. State relation LoanRequest

There is a request and response relation corresponding each task appearing in the workflow. They also have as a key attribute the workflow instance identifier (insId). In addition the request relations have an attribute for each input parameter of the task and the response relations have an attribute for each of its output parameters. The request and response relations corresponding to the tasks *Enter loan information* (ELI for short) and *Creditability* are presented in Figure 4.

RequestELI	insId	SSN	amount
	1003	5252	470
	1111	1234	1000

RequestCreditability	insId	SSN
	0012	3232

Fig. 4. Relations RequestELI and RequestCreditability

A workflow user activates an instance of the *Loan Request* workflow by passing through the *workflow execution interface* the attributes SSN (say 1234) and the amount of the loan (say 1000). The *workflow execution interface* assigns an unique identifier for the request (say 1111) and activates a workflow instance by the following SQL-insertion statement:

```
INSERT INTO LoanRequest(insId, SSN, amount)
  VALUES(1111, 1234, 1000)
```

From the workflow state relation in Figure 3 we can see that there are three workflow instances. The instance having the insId 0012 has already committed (i.e., has the value 'C') the first task but neither of the tasks *ELI* or *Liability* is committed (i.e., has the value '–'). The other two instances have been activated but they have not finished any task. From the relation RequestELI, in Figure 4, we can see that they both have an execution request on the task *ELI*. Note that as the instance 0012 has already committed the task *ELI*, it cannot have any request on the task *ELI*. Instance 0012 has a request on the task *Creditability* as it has a tuple in the relation *RequestCreditability* (Figure 4).

specification The state relation has one trigger specified for each task that appears in the workflow. So, as there are four tasks in the workflow *LoanRequest*, the *trigger generator* has generated four triggers for it. The names of the triggers are composed of the term Req followed by the name of the task to be activated. Each responce relation is also associated with a trigger. The function of these triggers is to update the workflow state relation.

So, after a new tuple is inserted to the state relation the trigger ReqELI inserts a new tuple into the relation RequestELI. Then the *task runner* removes the request from the relation RequestELI, executes the task and inserts a tuple into the relation ResponseELI. This insertion activates the trigger ResELI which removes the tuple from the response relation ResELI and updates the tuple having the same insId value in the relation LoanRequest. The update changes the value of attribute S-ELI attribute to be either 'commit (C) or 'abort (A).

SQL3 specifications of the triggers ReqELI and ResELI are the following:

```
CREATE TRIGGER ReqELI
  AFTER INSERT ON LoanRequest
  REFERENCING NEW AS NewTuple,
  INSERT INTO  RequestELI
    SELECT insID SSN amount FROM LoanRequest
    WHERE insId=NewTuple.insId

CREATE TRIGGER ResELI
  AFTER INSERT ON ResponseELI
  REFERENCING NEW AS NewTuple,
  UPPDATE LoanRequest
     SET (S-ELI) =
        (SELECT S-ELI FROM ResELI
           WHERE insId=NewTuple.insId)
     WHERE insId=NewTuple.insId
```

Note that the above triggers do not have any condition part (WHEN-clause). The 'on update' trigger which activates the *Creditability* task (not presented here) has the condition part which checks that the task *ELI* has committed, i.e., the attribute S-ELI in the state relation *LoanRequest* has the value 'C'.

6.2 Isolation in the example workflow

The isolation problem of the *Loan request* is that the decision of granting a loan is based among other things on the client's existing loans which are retrieved in the task *Creditability*. A necessary isolation requirement is that the loans are known when a decision to grant or reject a new loan request is done, i.e., one client may at a time have at most one active loan request. By setting a restriction which states that there may not be two tuples having equal values on the attribute SSN would solve the problem of concurrent loan requests. This can be done by specifying the attribute SSN to be UNIQUE or by inserting the following SQL-clause:

```
ALTER TABLE LoanRequest ADD (
  CONSTRAINT LoanWatch UNIQUE (SSN))
```

Though naming of constraints is not necessary in SQL, we give a name for each constraint as it simplifies the altering of constraints.

6.3 Options in the example workflow

An atomicity problem of the *Loan request* is that the liability evaluated in the task *Liability* should be valid when the decision to grant or reject the loan is done. If there are more than one concurrent instances then each may have seen the same liability and decide that the liability limit will not be exceeded if the loan is granted. However, granting more than one loan may cause that the liability limit is exceeded. To avoid such problem the *Liability* task is executed as an option. That is, the actual *Liability* task is executed as the preliminary task, which decreases the liability limit by the requested amount of the loan. If the loan is granted, then the positive complementary task increases the liability limit and the liabilities by the amount of the loan. The negative complementary task is executed, if the loan is not granted. It removes the option by increasing the liability limit by the amount of the requested loan.

7 Conclusions

The focus in the WorkMan project has been on investigating different approaches to utilize the services of the underlying database management systems in developing a workflow management system. Particularly the feasibility of SQL-clauses in specifying and implementing workflow executions, and supporting workflows' transactional properties are investigated.

Currently we are implementing the WorkMan system in a centralized environment based on the services of the Oracle database management system. Later on the system will be extended to support the interoperability of autonomous and distributed systems. This extension will be based on the standards of distributed computing.

A problem for us as well as with any model using semantic information is that workflow designer is burdened with writing options and restrictions. However, this threshold of this additional function is lowered as it can be done using SQL-features familiar to database designers. However, the only way to evaluate definitively the system is to test it in the case of some real application.

References

1. P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.
2. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Deriving active rules for workflow enactment. In *Proceedings of the DEXA'96*, 1996.
3. U. Dayal, M. Hsu, and R. Ladin. Organizing long-running activities with triggers and transactions. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 204–214, 1990.
4. A. Elmagarmid, Y. Leu, W. Litwin, and M. Rusinkiewicz. A multibase transaction model for interbase. In *Proc. of the 16th International Conference on VLDB*, pages 507–518, 1990.
5. H. Garcia-Molina. Using semantic knowledge for transaction processing in a distributed database. *ACM Transactions on Database Systems*, 8(2):186–213, June 1983.
6. H. Garcia-Molina and K. Salem. SAGAS. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 249–259, 1987.
7. H. Laine and J. Puustjärvi. Modeling business processes as transactional workflows. In *Proc. of the Workshop on Practical Business Process Modeling (in conjuction with CAiSE'00)*, 2000.
8. P.E. O'Neil. The escrow transactional method. *ACM Transactions on Database Systems*, 11(4):405–430, December 1986.
9. J. Puustjärvi. Negotiation transactions: An approach to increase the automation of workflows. In *Proc. of the 9th International Conference on Advanced Information Systems Engineering (CAiSE'97)*, pages 89–102, 1997.
10. J. Puustjärvi. *Transactional Workflows.* PhD thesis, University of Helsinki, 1999.
11. A. Schwenkreis and A. Reuter. The impact of concurrency control on the programming model of ConTracts. In *Proc. of the International Workshop on Advanced Transaction Models and Architectures (ATMA)*, 1996.
12. J. D. Ullman and J. Widom. *A First Course in Database Systems.* Prentice Hall, 1997.
13. H. Wächter and A. Reuter. The ConTract model. In A.K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, chapter 7, pages 219–263. Morgan Kaufmann Publishers, 1992.
14. J. Widom and Ceri. S. *Active Database Systems.* Morgan Kaufmann, 1995.

Formalizing Workflows Using the Event Calculus

Nihan Kesim Cicekli and Yakup Yildirim

Department of Computer Engineering, METU, Ankara, Turkey
{nihan,yakup}@ceng.metu.edu.tr

Abstract. The event calculus is a logic programming formalism for representing events and their effects especially in database applications. This paper presents the use of the event calculus for specifying and simulating workflows. The proposed framework maintains a representation of the dynamic world being modeled on the basis of user supplied axioms about preconditions and effects of events and the initial state of the world. The net effect is that a workflow specification can be made at a higher level of abstraction. Within this framework it is possible to model sequential and concurrent activities with synchronization when necessary. It is also possible to model agent assignment and concurrent workflow instances.

1 Introduction

A workflow management system provides support for modeling, executing and monitoring the activities in a workflow. There are many commercial products to model and execute workflows [e.g. 1,3,13,14,19] but it has been realized that a formal specification model is required for the analysis and reasoning about the workflows. The most common frameworks for specifying workflows are control flow graphs [10], event-condition-action rules [2,8] and temporal constraints[17,18]. There is also a logic-based formalism which proposes a concurrent transaction logic for specifying, analyzing and scheduling of workflows [7].

In this paper we propose a framework for specifying and executing workflows based on the Event Calculus [11]. The Event Calculus provides a framework for temporal reasoning by using the first-order predicate logic. We argue that the specification of workflow processes can be enhanced by integrating them with temporal databases and/or with temporal logic programming systems. Addition of a temporal dimension to workflows can enhance their facilities in different ways. It will provide mechanisms for storing and querying the history of all processes. This may serve the need for querying some piece of information in the process history. Or it may serve the need for mining the history of the workflow to analyze and assess the efficiency, accuracy and the timeliness of the processes.

In [4] we proposed a formulation of the event calculus to describe simple workflow specifications where one activity follows another in a sequential manner. In this paper we investigate the ways in which the event calculus can be used as a basis for more complicated workflow definitions where concurrent activities, agents and concurrent workflow instances can also be modeled. We show how an extended version of the event calculus can be used to model a workflow enhanced with temporal reasoning.

The rest of the paper is organized as follows. Section 2 reviews the workflow process definition by the Workflow Management Coalition. Section 3 summarizes the basics of the event calculus. The modeling of workflow processes using the event calculus is described in Section 4. The proposed framework is extended to include

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 222-231, 2000.

workflow manager, agents and concurrent workflows in Section 5. The computational issues are discussed in Section 6. We conclude the paper by summarizing the features of the proposed system in Section 7.

2 An Overview of Workflow Management

Workflow Management Coalition (WfMC) defines a 'reference model' which describes the major components and interfaces within a workflow architecture [20]. In a workflow, activities are related to one another via flow control conditions (transition information). According to this reference model we identify four routings among the activities:

1. *Sequential*: Activities are executed in sequence (i.e. one activity is followed by the next activity.)
2. *Parallel*: Two or more activities are executed in parallel. Two building blocks are identified: (a) AND-split and (b) AND-join. The AND-split enables two or more activities to be executed concurrently after another activity has been completed. The AND-join synchronizes the parallel flows, one activity starts only after all activities in the join have been completed.
3. *Conditional*: One of the alternative activities is executed. In order to model a choice among two or more alternatives two blocks can be used: (a) XOR-split and (b) XOR-join. Here no synchronization is required.
4. *Iteration*: It may sometimes be necessary to execute an activity or a set of activities multiple times.

In the rest of the paper, we discuss how these features of a workflow management system can be modeled in the framework of the event calculus. We first give a brief summary of the event calculus.

3 Event Calculus

The event calculus is a logic programming formalism for representing events and their effects, especially in database applications [11]. A number of event calculus dialects have been presented since this original paper[5,6,9,12,15,16]. The one described here is based on a later simplified version presented in [12].

The event calculus is based on general axioms concerning notions of events, properties and the periods of time for which the properties hold. The events initiate and/or terminate periods of time in which a property holds. As events occur in the domain of the application, the general axioms imply new properties which hold true in the new state of the world being modeled, and infer the termination of properties which no longer hold true from the previous state.

The main axioms used by the event calculus to infer that a property holds true at a time are as follows:

holds_at(P, T) ←
    happens(E, T1) , *T1 < T* , *initiates(E, P), not interrupted(P, T1, T).*
interrupted(P, T1, T2) ←
    happens(E', T'), terminates(E', P), T1 < T', T' <= T2.

The predicate *holds_at(P, T)* represents that property *P* holds at time *T*. The predicate *happens(E, T)* represents that the event *E* occurs at time *T*. The time points are ordered by the usual comparative operators. The formula *initiates(E, P)* represents that the event *E* initiates a period of time during which the property *P*

holds, and *terminates(E, P)* represents that the event *E* terminates any ongoing period during which property *P* holds. The *not* operator is interpreted as negation-as-failure. The use of negation-as-failure gives a form of default persistence. The formula *interrupted(P, T1, T2)* represents that the property *P* ceases to hold at some time between *T1* and *T2* due to an event which terminates it. The problem domain is captured by a set of *initiates* and *terminates* clauses. A particular course of events that occur in the real world being modeled is represented by using the predicate *happens*.

We want to use the event calculus in the specification of a workflow process. A workflow process contains a collection of activities and the order of activity invocations or conditions under which activities must be invoked (i.e. control flow) and also data flow between the activities. In the event calculus framework, events will denote the start and end time points of activities. The state of the workflow will be described by the properties. In other words events will specify the control flow and the effects of the events are used to describe the data flow within the workflow.

4 Specification of Workflow Processes in the Event Calculus

This section presents the event calculus as a first-order formalism for the specification of workflow processes. Once the event occurrences until time *t* are known, the state of the system can be computed at any point of time until *t*. Thus modeling can be regarded as the computation of event occurrences.

Each activity is initiated by an event and its termination is regarded as an event that starts one or more activities. Thus each activity ***A*** has a starting event ***start(A)*** and a last event ***end(A)***. Between these two events there may be several other sub-events defined for the activity.

4.1 Sequential Activities

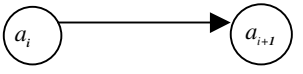


Fig. 1. Activity a_{i+1} starts when a_i finishes.

Figure 1 shows a graphical representation of sequential routing of activities. Circles represent an activity. When the activity a_i finishes, the next activity a_{i+1} starts. This can be formulated in the event calculus as follows:

$$happens(start(a_{i+1}), T) \leftarrow happens(end(a_i), T). \quad (1)$$

When the last event of the activity a_i happens (e.g. commit) the starting event of the next activity is triggered. In this rule when the event a_i commits the event a_{i+1} starts immediately (at the same time point *T*). This can be modified if necessary by delaying the start time of the next activity by a specific time period. For instance, if a_{i+1} starts after *x* time units we can write:

$$happens(start(a_{i+1}), T2) \leftarrow happens(end(a_i), T1), T2 \text{ is } T1 + x. \quad (2)$$

An activity begins executing when its start event occurs. It may have a predefined duration or its execution time period may depend on some conditions or occurrences of other sub-events. We use the term sub-event to refer to the events occurring within the activity. Thus, the last event of an activity can happen either after some predefined time or after some condition is met. This and some other sequencing among activities can be described in the event calculus as follows:

happens(end(a), T) ← *happens(start(a), T1), T = T1 + y .* (3)
happens(end(a), T) ← *happens(a_{sub}, T) .* (4)
initiates(start(A), active(A)). (5)
terminates(end(A), active(A)). (6)

In rule (3) *y* is a constant used to denote the predefined time period between events *start(a)* and *end(a)*. This value can be given statically at the time of the specification of the workflow or it may be determined by the current state of the workflow. Or as in rule (4), the end of an activity may be determined by the occurrence of a sub-event (a_{sub}) within the activity. Rules (5) and (6) show how the effects of starting events can be represented. For instance rule (5) states that, if the starting event of an activity happens, it initiates the time period for which that activity is active. The property *active(a)* holds between the happening times of the events *start(A)* and *end(A)*.

4. 2 Concurrent Activities

In a workflow, some activities are executed concurrently. Figure 2.a illustrates AND-split. When the activity a_i finishes, activities a_1, a_2 … a_n start concurrently. Figure 2.b illustrates AND-join. Here the activity a_j starts when all the preceding activities finish.

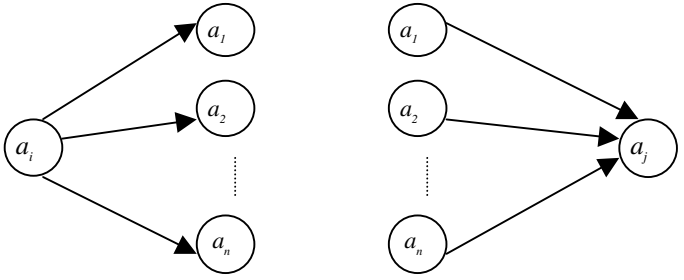


Fig. 2. (a) AND-split (b) AND-join

In the event calculus we represent AND-split with a sequence of rules:

happens(start(a_1), T) ← *happens(end(a_i), T).* (7)
happens(start(a_2), T) ← *happens(end(a_i), T).*
…
happens(start(a_n), T) ← *happens(end(a_i), T).*

The following rule is used to represent an AND-join of activities:

happens(start(a_j), T) ← (8)
happens(end(a_1), T_1), happens(end(a_2), T_2),…, happens(end(a_n), T_n),
T = max(T_1, T_2, …, T_n).

Activity a_j waits for the completion of all activities a_1 .. a_n. The last conjunct in rule (8) is to start the activity at the time of the last ending activity among activities a_1 .. a_n.

4.3 Conditional Activities

In a workflow the execution of some activities may depend on certain conditions. If the conditions are satisfied those activities are executed; otherwise they are not executed. This kind of execution specification can be represented in a similar fashion as in Fig. 2, except that the arrows are labeled with conditions. (Fig. 3.a)

Such transitions can be used to describe XOR-splits. In these transitions only one of the alternative activities is executed depending on the evaluated condition. The

important point here is that the conditions must be exclusive. In other words only one of the conditions should be true at the time of the decision in order to guarantee that only one execution path is chosen.

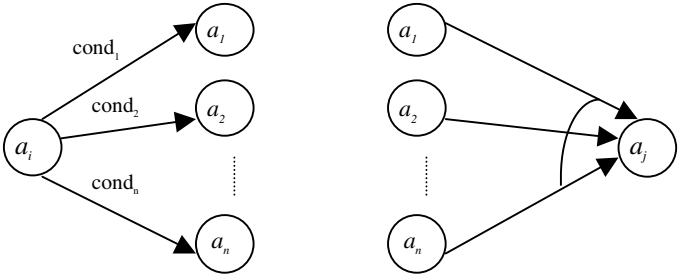


Fig. 3. a) XOR-split b)XOR-join

We can represent the conditional activities in the event calculus as follows:

$happens(start(a_1), T) \leftarrow happens(end(a_i), T), holds\_at(condition_1, T).$ (9)
$happens(start(a_2), T) \leftarrow happens(end(a_i), T), holds\_at(condition_2, T).$
…
$happens(start(a_n), T) \leftarrow happens(end(a_i), T), holds\_at(condition_n, T).$

When the activity a_i ends, the activity a_1 or a_2 …or a_n starts according to the condition satisfied at that time. The conditions may be a state check (i.e. a *holds_at* predicate) or another predicate *happens* checking the occurrence of another event. Only one of the conditions must become true at the time of the execution. If none of the conditions is satisfied at the end of the activity a_i, then the following activity cannot start. In order to prevent this problem an additional path can be provided as an "otherwise" option. This can be represented in the event calculus as follows:

$happens(start(a_{n+1}), T) \leftarrow happens(end(a_i), T),$
$not\ happens(start(a_1), T), not\ happens(start(a_2), T), \ldots, not\ happens(start(a_n), T).$

In XOR-join (Fig.3b) if any one of the incoming activities is finished, the activity at the join can start executing. Here there is no need for the synchronization of the incoming activities. The completion of one of the incoming activities is sufficient to start the joined activity. The description of XOR-join in the event calculus needs some care. It may be seen as simply listing *n* rules in the form:

$happens(start(a_j), T) \leftarrow happens(end(a_k), T).$

for each $k = 1,\ldots, n$. However this formulation will not yield the desired execution, because the follow-up activity a_j will be started every time an incoming activity is finished (i.e. *n* times). We have to ensure that a_j is started only once. In order to achieve this we represent the XOR-join by the following rules:

$happens(start(a_j), T) \leftarrow$ (10)
$happens(end(a_1), T), endsameorafter(a_2,T), \ldots, endsameorafter(a_n, T).$
$happens(start(a_j), T) \leftarrow$ (11)
$happens(end(a_2), T), endafter(a_1,T), endsameorafter(a_3,T), \ldots,$
$endsameorafter(a_n, T).$
…
$happens(start(a_j), T) \leftarrow$ (12)
$happens(end(a_n), T), endafter(a_1,T), endafter(a_2,T), \ldots, endafter(a_{n-1},T).$

Rule (10) states that the end of the activity a_1 starts a_j only if the activities $a_2 \ldots a_n$ end at the same time as a_1 or after. Rule (11) states that the end of the activity a_2 starts

a_j only if the activities listed before a_2 (i.e. a_1) end after a_2 and the activities listed after a_2 (i.e. $a_3 \ldots a_n$) end at the same time or after a_2. Finally, rule (12) represents that the end of the activity a_n starts a_j if the activities listed before a_n end after a_n. The predicates *endsameorafter(A, T)* checks if the activity *A* ends at the time instant *T* or after. Similarly, the predicate *endafter(A, T)* checks if the activity *A* ends strictly after time *T*. These predicates are defined as follows:

endsameorafter(A, T) ← *happens(end(A), T1), T ≤T1.*
endsameorafter(A, T) ← *not happens(end(A), _).*

endafter(A, T) ← *happens(end(A), T1), T < T1.*
endafter(A, T) ← *not happens(end(A), _).*

The second clauses are used to handle the case that the end of the activity has not happened yet (i.e. until time *T*).

4.4 Iteration

Sometimes we may need to execute a group of activities one or more times. This loop may be executed for a certain number of times. (Fig.4.)

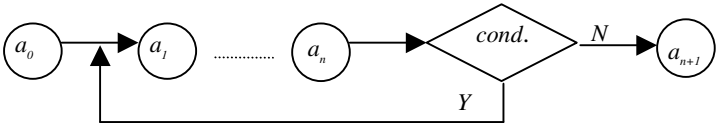


Fig. 4. Activities a_1 to a_n are executed several times.

In the event calculus, we may describe the iteration of activities as follows:

happens(start(a_1), T) ← *happens(end(a_0), T).* (13)
happens(start(a_1), T) ← *happens(end(a_n), T), holds_at(loopcondition, T) .*
happens(start(a_{n+1}), T) ← *happens(end(a_n), T), not holds_at(loopcondition, T).*

The activities between the activities a_1 and a_n can be arranged in any of the transition types that we mentioned before. The number of the iterations can be controlled with a loop count variable that is increased by the last event of the iteration or with any condition that checks the state of the system by the time of the last event in the iteration.

5 Workflow Management

A workflow involves several activities, each performed by an agent – human or automated. A workflow management system consists of a scheduler (manager) agent and task agents. A task agent is responsible for the execution of an activity. The workflow manager is an agent that coordinates the execution of all activities according to the workflow specification. It knows which activities follow which ones under what conditions. When an activity is to be executed, the manager assigns an agent that will execute that activity. Thus the manager must keep track of available agents as well. This section discusses how the manager design is done within the event calculus.

5.1 Agent Assignment

Each agent can perform one or more activities; and each activity can be executed by one or more agents. Which agents can perform which activities is specified with the predicate *qualified(Agent, Activity)*. When an activity is to be executed, an agent that is qualified for that activity is selected and the activity is assigned to that agent if the agent is idle. If the agent is not idle either another available, qualified agent is selected or the activity is kept waiting for the agents to finish their jobs. For instance, in a sequential execution of activities (Fig.1), the assignment of the agent is expressed by the following rule:

happens(assign(Agent, a_{i+1}), T) ← (14)
happens(end(a_i, _), T),
qualified(Agent, a_{i+1}), holds_at(idle(Agent), T),
not anysmalleragent(Agent, a_{i+1}, T).

anysmalleragent(Agent, A, T) ← (15)
qualified(Agent2, A), Agent2 < Agent,
holds_at(idle(Agent2), T).

Rule (14) represents that the activity a_{i+1} is assigned to the qualified agent *Agent* at the completion of the activity a_i only if *Agent* is idle at that time and there is no other agent with a smaller number can run the activity. We must ensure that the same activity is not assigned to more than one agent at the same time. We can achieve this by ordering the numbers of agents and checking for the minimum numbered idle agent to assign the activity to (rule 15). Here, the number used in the comparison of agents is an abstraction. It may denote the cost of executing the activity on the agent or it may denote a priority in assigning the activity to an agent in the definition of the predicate *qualified.*

When an agent *Agent* is assigned to an activity *A*, it starts executing that activity *A*. This can be represented by the following rule:

happens(start(A, Agent), T) ← *happens(assign(Agent, A), T).* (16)

Here, we specify the agent explicitly in the event name.

When an activity starts being executed by an agent, the agent is not idle any more and it is assigned to that activity until it finishes the activity. When the activity is finished the agent is released and it becomes idle again, ready to execute the next activity. We describe these two states of an agent with two predicates: *idle(Agent)* and *assigned(Agent, Activity)*. The state of the agent may be changed by two events: *assign(Agent, Activity)* and *release(Agent, Activity).* Thus, we write the following rules:

initiates(assign(Agent, Activity), assigned(Agent, Activity)).
initiates(release(Agent, Activity), idle(Agent)).

terminates(assign(Agent, Activity), idle(Agent)) .
terminates(release(Agent, Activity), assigned(Agent, Activity)).

happens(release(Agent, Activity),T) ← *happens(end(Activity, Agent) , T).*

There are cases in which a task needs an agent but all agents are busy. In this case the activity must wait until one of the agents becomes idle. For instance, assume that in a sequential routing, after the last event of an activity a_i, the manager looks for an idle agent to execute activity a_{i+1}. If there is no idle agent, activity a_{i+1} starts waiting for an appropriate agent:

initiates(end(a_i,_), waiting(a_{i+1}, Agent, T)) ← (17)
happens(end(a_i, _), T), qualified(Agent, a_{i+1}),
holds_at(assigned(Agent, C), T), C ≠a_{i+1},
not anyotheragent(Agent, a_{i+1}, T).

anyotheragent(Agent, a_{i+1}, T) ← (18)
qualified(Agent2, a_{i+1}), Agent ≠Agent2,
holds_at(idle(Agent2), T).

The activity a_{i+1} will be kept waiting if all the agents qualified for this activity are assigned to other activities. Rules (17) and (18) check the availability of all the qualified agents for a given activity.

It is also possible that several activities may wait for the same agent. If the agent becomes idle then the activity that has waited longest (i.e. the one with the smallest timestamp) is assigned to that agent (rule 19). This rule also deals with the problem of having more than one agent becoming idle at the same time. If that is the case, the activity is assigned to the minimum numbered agent:

happens(assign(Agent, Act), T) ← (19)
happens(release(Agent, _), T),
holds_at(waiting(Act, Agent, T1), T),
not waitinglonger(Act, Agent, T1, T),
not releasedotheragent(Agent, Act, T).

waitinglonger(Act, Agent, T1, T) ← (20)
holds_at(waiting(Act2, Agent, T2), T), Act ≠Act2, T2 < T1.

releasedotheragent(Agent, Act, T) ← (21)
qualified(Agent2, Act), Agent2 < Agent,
happens(release(Agent2, _),T).

Rule (20) checks for any other activity that has waited longer for the currently released agent. Rule (21) checks for any other qualified agent being released at the same time. When an activity is assigned to an agent that activity does not wait for any agents any more (rule 22).

terminates(assign(Agent, Act), waiting(Act, _, T1)) (22)

5.2 Concurrent Workflow Instances

We have so far represented how to specify workflow activities and execute a single instance of a defined workflow within the event calculus framework. However in an application, there can be several instances of the same workflow executing at the same time. The available agents can perform the activities of any of the workflow instances. Thus the problem is how to represent more than one workflow instance concurrently within the current framework.

In order to model concurrent workflow instances, we number the workflow instances. Each workflow instance is assigned a unique number and every activity in a workflow is associated with a workflow instance. Thus we add the number of the workflow instance to the rules. For instance in the following rule we specify a sequential routing of activities using workflow instances:

happens(start(a_j, Wno), T) ← *happens(end(a_i, Wno), T).* (23)

When an activity a_i in a workflow instance *Wno* finishes the next activity in the same workflow starts. In this rule we omit the agent assignment for simplicity. If concurrent workflow instances are modeled, the number of the workflow instance must be added to all the previous rules that we have discussed earlier.

6 Implementation Issues

The manager actually runs the workflow specification (rules 1-13). The manager starts a workflow process when an initial event is happened. Once that is recorded, the manager starts the other activities according to the workflow definition and the available agents. In our framework the workflow manager has a centralized control over the agents. All the agents inform the manager agent of the end of the activity (by *end(A)*) when they finish it. The manager starts an activity at an agent by recording the event *assign(Agent, Act)*. If no available agents are found for an activity, then the activity is kept waiting. When an agent is released one of the waiting activities is assigned to that activity.

We have given rules to describe the task of the manager agent with sequential routing of activities (rules 14-23). These rules can be applied to other routings of the activities that were specified in Section 4 in a similar way.

The theory can be implemented in several different ways. One approach is to write the axioms more or less directly in Prolog. However a direct translation of the axioms into a Prolog program will cause an infinite loop, because the definition of *holds_at* includes calls to *happens* and that in turn includes calls to *holds_at.*

We have overcome this problem by rewriting the axioms so that they are more suitable for SLDNF resolution (but perhaps less declarative) [4]. In order to achieve this we consider the causality of events. We rewrite the clauses so that a Prolog interpreter can proceed forwards in time from the earliest known event, maintaining a list of ongoing events. Since we know which events occur after which events, we can compute the entire history given the initial event(s). We proceed roughly in a bottom-up manner: we compute what events the initial events cause in the history, then compute what events these cause in the history, and so on. The same approach is used in the implementation of the currently proposed extended framework.

7 Conclusion

We have demonstrated how the event calculus might be extended to describe the specification and execution of activities in a workflow. In [4] we only demostrated the use of the Event Calculus in modeling sequential workflows. In this paper we have shown that other major types of activity routings in a workflow (namely sequential, parallel, iterative, conditional) can also be expressed using the axioms of the event calculus in a declarative way. We have also illustrated that agent assignment and concurrent workflow instances can be modeled in the event calculus.

The proposed framework can be used as a quick tool in prototyping applications and/or simulations. Due to its additional temporal dimension, it provides facilities for querying the history of all activities, thus providing opportunities to analyze the efficiency of the workflows.

The current framework is currently being extended to include exception handling in the workflows. We also investigate the ways of modeling transactional workflows in which rollback and compensation activities can also be specified using declarative rules.

References

1. Alonsa, G., D. Agrawal, A. El Abdabi, and C. Mohan. Functionalities and Limitations of Current Workflow Management Systems. In *IEEE-Expert (Special Issue on Cooperative Information Systems)*, 1997.
2. Baral C., J. Lobo. Formalizing workflows as collections of condition-action rules. Dynamics '97, Workshop in ILPS, 1997..
3. Barbara, D., S. Mehrotra and M. Rusinkiewicz. INCAs: Managing Dynamic Workflows in Distributed Environments. *In Journal of Database Management*, vol.7, no.1, 1996.
4. Cicekli-Kesim, N. A Temporal Reasoning Approach to Model Workflow Activities, Lecture Notes in Computer Science, no. 1649, ed. R.Y. Pinter and S. Tsur, NGITS'99, Zikhron-Yaakov, Israel, July 1999.
5. Kesim, N. and M. Sergot. A Logic Programming Framework for Modelling Temporal Objects. In the IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 5, October 1996.
6. Kesim, F.N. and M. Sergot. Implementing an Object-Oriented Deductive Database Using Temporal Reasoning. In the Journal of Database Management, Vol. 7, No. 4, 1996.
7. Davulcu, H., M. Kifer, C.R. Ramakrishnan, and I.V. Ramakrishnan. Logic Based Modeling and Analysis of Workflows. In *ACM Symposium on Principles of Database Systems*; Seattle, Washington, ACM Press, 1998.
8. Dayal, U., M.Hsu and R. Ladin. Organizing Long_Running Activities With Triggers and Transactions. *In ACM SIGMOD Conference on Management of Data*, 1990.
9. Evans C. The Macro-Event Calculus: Representing Temporal Granularity. Technical Report, Imperial College, London, 1989.
10. Harel, D. StateCharts: A Visual Formalism for Complex Systems. Science of Computer Programming, 8:231-274,1987.
11. Kowalski, R.A. and M. J. Sergot. A Logic-based calculus of events. *New Generation Computing*, 4, pp. 67-95, 1986.
12. Kowalski, R.A. Database Updates in the Event Calculus. *Journal of Logic Programming* 12(1-2): 121-146 (1992).
13. Krishnakumar, N. and A. Sheth. Managing Heterogeneous Multi-System Tasks to Support Enterprise-wide Operations. In *Distributed and Parallel Databases*, Vol.3, No.2, April 1995.
14. Mohan, C., G. Alanso, R. Gunthor and M. Kamath. Exotica: A Research Perspective on Workflow Management Systems. In *Data Engineering*, Vol.18, No.1, March 1995.
15. Shanahan, M. Representing Continuous Change in the Event Calculus. *ECAI*, pp. 598-603, Stockholm, Sweden, 1990.
16. Shanahan, M. A Simple Logical Framework for Prediction Problems, Technical Report, Logic Programming Group, Imperial College, November 1988.
17. Singh, M.P. Semantical Considerations on Workflows: An algebra for Intertask Dependencies. *In Proceedings of the International Workshop on Database Programming Languages*, Gubbio, Umbria, Italy, September 6-8 1995.
18. Singh, M.P. Synthesizing Distributed Constrained Events From Transactional Workflow Specifications. In *proceedings of 12-th IEEE Intl. Conference on Data Engineering* , pp. 616-623, New Orleans, LA, February 1996.
19. Wachter, H. and A. Reuter. The ConTract Model. *In Transaction Models for Advanced Database Applications*, Chapter 7, Morgan_Kaufmann, February 1992.
20. Workflow Management Coalition. Terminology and Glossary. Technical Report (WFMC-TC-1011), Workflow Management Coalition, Brussels, 1996.

Gaining Control over Time in Workflow Management Applications

Eleanna Kafeza Kamalakar Karlapalem

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{kafeza,kamal}@cs.ust.hk

Abstract. The need for explicit time management in workflow environments has been recently identified. Although the concept of time is inherent in workflow applications, time management until now has been treated by the same general-purpose structures of the workflow system. This traditional approach does not allow for explicitly specifying timing correctness requirements, temporal consistency checking, immediate control over the diverse set of time constraints that workflow applications exhibit and timely monitoring of the environment. In this paper, we extend the existing workflow specification based on the requirements posed by real-life applications such as health systems. We argue that time management should be an integral part of the workflow management system and not performed by a general-purpose temporal reasoner. We incorporate a subset of interval algebra that allows for efficient consistency checking, while providing expressiveness of temporal constraints. We show that static scheduling for meeting temporal constraints is inadequate for a large class of workflow applications. We show how global scheduling based on temporal constraints can be combined with agent scheduling policies. We demonstrate through examples the working of the scheduling algorithms.

1. Introduction and Motivation

The need of explicitly managing time in workflow management systems has been recently identified [ZS-99,AM-97,C-97,EPR-99,H+-96]. One of the limitations of workflow systems is that they lack the expressiveness for specifying temporal relationships across activities. Therefore, they fail to provide adequate timing control during the execution. Although work has been done to specify and enforce different types of coordination of activities during workflow execution [RS-94, KR-98, ASSR-93], none of them concentrates explicitly on time-management, offering thus a specialized and efficient control over time during the workflow execution. According to [H+-96] there has been a misconception regarding the significance of temporal reasoning in workflow environments and how they can be utilized. The main objective of this paper is to develop a framework for specifying temporal constraints, provide the means for consistency checking of the specification, and provide scheduling mechanisms that induce time-based execution of workflow.

M. Ibrahim, J. Küng, N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 232-241, 2000.

In [EPR-99], the authors model time-constraints as lower and upper bound constraints that specify specific time distances among event occurrences. In our framework, we do not deal with distance-based constraints. We provide a wide set of temporal constraints between activities by exploiting all possible orderings of start and completion events between two activities. We allow a wider range of dependencies to be translated into specific temporal ordering of activities therefore we incorporate temporal aspects into workflow execution from a different perspective. Among the research prototypes, ADEPT [DR-98] incorporates temporal aspects. However, its functionality is restricted in checking whether minimal and maximal time distances between tasks are kept. In [ZS-99], the authors identify the need for temporal workflow management when facing claim-handling situations. They present time allocation and tasks prioritization policies. Although they address a different problem, they do identify the need for efficient management of time in workflow applications. We motivate our approach based on existing examples for health systems [H+-96].

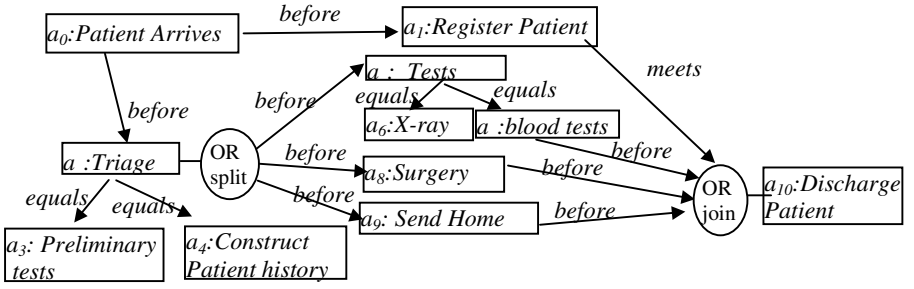


Fig. 1. Emergency room example

The above example describes an emergency room process (see Figure 1) [H+-96]. Several temporal requirements restrict the relative order of activity execution. Moreover, due to the OR constructs we cannot schedule the workflow based on a pre-defined decision as described in [H+-96]. There is a diverse set of application domains such as health systems, spatial database applications [AH-97], inter-organizational workflows [WfMc] etc, where conventional temporal controls are inadequate to offer the required advanced temporal coordination among activities. In order to alleviate this problem we propose the incorporation of temporal constraints in the workflow specification. In this paper:

1. we present a framework the T-WfMc model that allows for the specification of temporal workflows and the corresponding WFMS architecture
2. we present a mechanism for consistency checking of the specification
3. we present scheduling algorithms under different perspectives

The organization of the paper is as follows: section 2 is the time-constrained workflow process model, section, in section 3 we examine the consistency of the specification, in section 4 we present the architecture and in section 5 we propose scheduling policies. Section 6 is the conclusions.

2. T-WfMC Model for Temporal Constraints

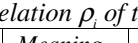
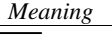
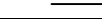
An *activity* is an atomic unit of work that forms a logical step within the workflow process. An activity can be in one of the following states*: inactive*, *active*, *suspended* and *completed*. An event is an abstraction that characterizes significant changes of an activity state during a workflow instance execution. In our framework, we consider two types of events: *start* events that signify the transition of an activity from *inactive* to *active* state and *complete* events that signify the transition of an activity from *active* to *completion* state. We assume a function *time* (*e*) that returns the time of occurrence of event *e*. Note that all the organizations adhere to a global clock whose variance does not affect the synchronization and scheduling of the workflow.

Definition: *An activity is a tuple* $(a_i, [s_i, c_i], d_i, dur_i)$ *such that* a_i *is the activity name that uniquely identifies the activity,* s_i *is activity's start time,* c_i *its completion time,* d_i *its deadline and* dur_i *its duration.*

Parameters s_i and c_i are calculated by the function *time*( ) applied to *start* and *completion* events of the activity. For example, given an activity a_i, $time(start(a_i)) = s_i$ and $time(complete(a_i))= c_i$. If the state of the activity is *inactive,* s_i and c_i take value $\propto$ denoting that the activity has not been active yet. If the state of the activity is *active*, s_i is the time the activity started execution and c_i is set to $\propto$. If the state of the activity is *completed*, s_i and c_i are the times the activity started and completed execution, respectively. We call the time interval $[s_i, c_i]$ when the activity is active, *active interval of* a_i.

Given two activities a_i and a_j and their corresponding *active intervals* $[s_i, c_i]$ and $[s_j, c_j]$, we need seven relations to associate them [A-83]: *before*, *meets*, *overlaps*, *starts*, *covers, finishes*, *equal* (TC = {**b**, **m**, **o**, **s**, **c**, **f**, **eq**}, respectively, see table 1).

Definition: *Given two activities* a_i *and* a_j *we say that they are constrained by relation* $\rho_i \in TC$ *denoted by* $\rho_i(a_i,a_j)$, *iff their active intervals* $[s_i,c_i]$ *and* $[s_j,c_j]$ *satisfy the constraints of relation* ρ_i *of table 1.*

Relation	*Meaning*	*Constraint*
a_i *before* a_j		$s_i<c_i<s_j<c_j$
a_i *meets* a_j		$s_i< c_i=s_j<c_j$
a_i *overlaps* a_j		$s_i<s_j<c_i< c_j$
a_i *starts* a_j		$s_i=s_j< c_i< c_j$

Relation	*Meaning*	*Constraint*
a_i *covers* a_j		$s_i<s_j< c_j< c_i$
a_i *finishes* a_j		$s_j <s_i<c_i= c_j$
a_i *equal* a_j		$s_i=s_j<c_i= c_j$

Table 1: Temporal relations and their corresponding constraints

For example, if activities a_i and a_j are constrained by the relation *covers*, then in every workflow instance execution their corresponding events should occur in the following temporal order: $start(a_j)$ $start(a_i)$ $complete(a_i)$ $complete(a_j)$.

Definition: *A time-constrained process specification is a graph P(A,E) where A is the set of nodes, and E the set of labeled edges such that the nodes of the graph represent the activities and a pair of activities is connected by an edge labeled* ρ_i *iff the activities execution is constrained by temporal relation* ρ_i.

The resulting graph is an *Interval Graph* [Vb-92]. Although we use seven out of the thirteen possible relationships [A-83] we can still express same set of constraints since whenever we want to specify inverse-$\rho_i(a_i,a_j)$ we can specify instead $\rho_i(a_j,a_i)$ (i.e. invert the direction of the arc in the graph).

3. Consistency of T-WfMC Workflow

In this section, we present consistency-checking techniques applied at build time and guarantee the consistency of the temporal specification. There are two types of consistency checking *constraint,* and *duration-deadline*. We say that the time-constrained process definition is *constraint consistent* iff there can be an execution that satisfies all the constraints. The *path consistency* technique [M-77] (the reader is referred to the references [DMP-90] for more details) is applied for verifying the 3-consistency of a given temporal network that allows labels with disjunction of temporal constraints. In our case, we use a subset of interval algebra that allows constraint consistency checking in $O(n^2)$ [Vb-92].

A constraint consistent specification is *duration-deadline consistent* iff there is an execution where activities last exactly as expected and complete their execution before their deadlines. We describe the duration and deadline consistency problem as a constraint satisfaction problem (CSP). Although a CSP problem in our case the complexity of the algorithm that finds a consistent solution is polynomial and efficiency is not sacrificed for verifying the correctness of specification.

The domain for activity a_i is denoted as $Domain(s_i)$ and represents the time *interval in which* the activity is eligible to start execution. Let a_i and a_j be two activities constrained by a temporal constraint $\rho_i(a_i,a_j)$. The procedure $REVISE(\rho_i(a_i,a_j), Domain(s_i))$ calculates the domain of activity a_j based on all possible active intervals of activity a_i and returns two Boolean variables DELETE and INCONSISTENCY. DELETE is true if the domain of the activity a_j is updated by the procedure and INCONSISTENCY is true if the duration of the activity a_j does not conform to the constraint specification (i.e., if the constraint is *equals* both activities should have the same duration). In the following example, we present *REVISE* for the *finish* constraint in detail. For the rest of the constraints we show the invariants. Note that $c_j = s_j + dur_j$. For *strict* constraints namely, *finish*, *equal*, *meets* and *starts* given a value for the start time of activity a_i there is only one value for c_j and *REVISE* is as follows:

- *REVISE* ($finish(a_i,a_j)$, $Domain(s_i)$:$[w,z]$):
  DELETE=false; INCONSISTENCY=false;
  if $dur_i < dur_j$ INCONSISTENCY=true
  $temp\_Domain(s_j)=[w+dur_i-dur_j, z+dur_i-dur]$ // $c_j=s_j+dur_j$
  if $current\_Domain(s_j) \neq temp\_Domain(s_j)$
  $current\_Domain(s_j)=current\_Domain(s_j) \cap temp\_Domain(s_j)$; DELETE=true;
  return [DELETE, INCONSISTENCY]
  end
- *REVISE* ($equal(a_i,a_j)$, $Domain(s_i)$): $s_j = s_i$, , $dur_j = dur_i$, *REVISE* ($meets(a_i,a_j)$, $Domain(s_i)$): $s_j = c_i$, and *REVISE* ($starts(a_i,a_j)$, $Domain(s_i)$): $s_i = s_j$, $dur_i < dur_j$

For *flexible* constraints namely, *before*, *overlaps* and *covers*, we can calculate a range of time interval denoting the earliest and the latest time the activity can start :

- *REVISE* (*before*(a_i,a_j), *Domain*(s_i)) : *Domain*(s_j)=[c_i+1,∞], *REVISE* (*overlaps*(a_i,a_j), *Domain*(s_i)) : *Domain*(s_j)=[s_i+dur_i–dur_j+1 , dur_i–1], $dur_j \geq 2$, $dur_i \geq 2$ and *REVISE* (*covers* (a_i,a_j), *Domain*(s_i)): *Domain*(s_j)=[s_i+1, c_i–dur_j–1] $dur_j < dur_i$.

```
Duration and Deadline Consistency Assurance algorithm- an
arc-consistency algorithm
Input: Time-constrained Process graph (TG), duration of
activities, and activities deadline
Output: determines duration and deadline consistency
DDCA-preprocessing (TG);
begin
 Domain(s_o)=0;  //initialization
 for i=1 to n do Domain(s_i)=[-∞, ∞];
 //insert arcs of TG in queue Q
Q← {ρ_i(a_i,a_j) | (a_i,a_j) ∈ arcs(TG) and ρ_i is the label of
(a_i,a_j), in topological sort};
end
DDCA( queue: Q)
 repeat
    begin
      CHANGE=false;
      ∀(a_i,a_j) ∈ Q: [DELETE, INCONSISTENCY]= REVISE(ρ_i(a_i,a_j),
                      Domain(s_i));
      if INCONSISTENCY= =true exit;
      //deadline violation
      ∀x∈ Domain(ts_j) : if x+dur_i>d_j CHANGE=true; Domain(s_j)=
      Domain(s_j)-x;
      CHANGE← (DELETE or CHANGE);
      //empty domain signifies inconsistency
      if Domain(ts_j)= =∅ INCONSISTENCY=true;  exit;
    end
 until ¬ CHANGE
```

Fig. 2. The DDCA Algorithm

All the constraints that restrict the activities are of the form $t_j=t_i+c$ where c is a constant (only two variables are allowed in the equalities). According to [HDT-92] for system of constraints with the above property (the constraints are basic arithmetic constraints), if it is *arc-consistent* then it is *satisfiable* (consistent). Therefore, for checking the consistency it suffices to apply an arc-consistency algorithm. If the domain of one variable becomes empty then the specification is inconsistent otherwise it is consistent. Notice that consistency means there is one solution that satisfies all the constraints. In Figure 2, we present the duration and deadline consistency algorithm (DDCA), which is an arc-consistency algorithm (AC-1) [M-77] configured to fit in our specific problem. The complexity of the algorithm is cubic.

The *REVISE* procedure for every two activities constrained by a temporal constraint eliminates those values that cannot be satisfied by the constraint. Although immediately after the application of *REVISE* the corresponding arc is consistent, it may not remain consistent because of changes made to the domain of one of the participating variables due to other constraints. Therefore, arc-consistency algorithm iterates until no more changes occur. Note that in the case of OR-constructs we

compute the "tighter" constraint in the sense that we assume that the worst-case path was followed.

Example: *In Figure 3, we present the results after applying the DDCA algorithm to the emergency room example.. The bold brackets in the graph denote the time at which an activity can start execution and still meet both deadlines and constraints. If we initiate the activities according to the left ends of these intervals and if activities last as specified then we have produced a consistent execution. Even more, this is the earliest possible execution. We depict the optimal execution with respect to workflow duration in the bottom figure of the example.*

a_i	d_i	dur_i
a_0	5	5
a_1	15	6
a_2	9	2
a_3	9	2
a_4	9	2
a_5	13	3
a_6	13	3
a_7	15	3
a_8	17	6
a_9	23	1
a_{10}	25	2

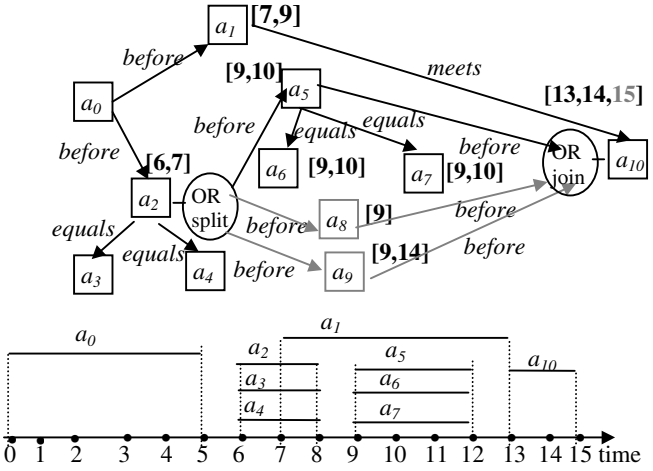


Fig. 3. DDCA on the emergency room example

4. System Architecture for T-WfMC

The time-constrained process definition is specified in the *process definition tool*. A graphical tool that allows the design of labeled graphs can do this. As shown in Figure 4, a separate *verification tool* is incorporated in the process definition tool. The verification tool is responsible for checking the consistency of the specification, by implementing the algorithms described in section 3. When an inconsistency is detected, the user is presented with the constraint that was violated and the user can modify the specification accordingly. Both process definition tool and verification tool communicate with the workflow database to acquire the necessary information about activities.

The workflow engine is responsible for the runtime control environment within the enactment service. One of its responsibilities is to handle navigation between activities (such as parallel/serial execution, and deadline scheduling). In the workflow engine, we incorporate a specific global scheduling tool. This tool consists of a set of global scheduling algorithms responsible for interacting with agents, submitting activities such that all temporal constraints are satisfied. Application agents are responsible for invoking applications, which can be located on a separate network

accessible platform. The global scheduler is also responsible for identifying inconsistencies and initiates exception-handling mechanism, if required. Notice that different workflow instances might have different global schedulers depending on the workflow application requirements.

The global scheduler communicates with the process definition tool to acquire information about the interval graph specification and agents for submitting activities for execution and receiving acknowledgments. The agent receives requests for initiating activities by the global scheduler, invokes necessary applications for automated tasks or inserts the request in the worklist of the role responsible for executing the activity.

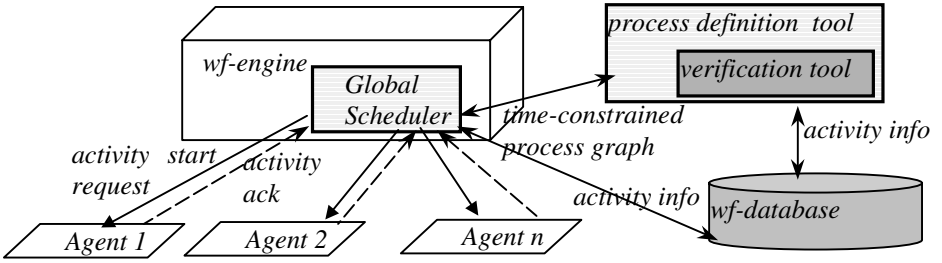


Fig. 4. Temporal Workflow Management System architecture

5. Scheduling T-WfMc Workflows

We address the problem of scheduling based on the requirements of the applications. We develop global scheduling algorithms so that constraints are maintained. The *true-agent pseudo-static scheduler* is applicable when activity duration is known and the agents are *true*. *True* agents are trustworthy -always start and complete activities when requested. Depending on the correctness requirement, we have *strict* and *soft* true-agent pseudo static scheduler:

a) Strict true agent pseudo static scheduler: this is the case where we are interested in *strict* correctness that is the workflow instance can commit only if we do not miss any temporal constraints and deadlines. The problem is that it is only at run time that we can decide which path of the OR constructs is taken and compute the correct start time for activities. Therefore, we have to compromise and take the worst-case execution, where at each time we assume that the "latest" branch will be followed. Since we have already executed the DDCA algorithm at build-time, we only have to take the right ends of the domains assigned to activities. This results in the latest possible consistent execution [HDT-92]. Note that one of the advantages of this approach is that there is no need for any computations at run-time.

The algorithm (Figure 5) iterates between OR-split constructs. It submits activities according to their latest initiation time until the next OR-split construct or End-of-graph is reached. When the activities have completed execution, it then decides upon the branch to be followed. From the agents' point of view, true-agents initiate activities at the time requested by the scheduler and they complete them at time start plus duration.

We assume that each node has a variable TYPE that denotes whether it has an associated OR-split or OR-join domain and a variable VISITED which is true if all the constraints associated to the current OR node have been visited. The first OR node of the workflow is defined as OR-split type and the last OR node as OR-join. Procedure *submit*(a_i, *right-end*(*Domain*(a_i)) requests from the corresponding agent the execution of activity a_i at time that is equal to the right end of the *Domain*(a_i).

```
Strict-True-Agent Pseudo-Static Scheduler:
Input: Time-Constrained Process Graph (TG)  with DDCA
assigned domains
STAPS(graph: TG)
begin
        repeat
       a_i = next_node(TG);
       if ai.TYPE == activity insert(a_i,Q);
       if a_i.TYPE == OR-split  evaluate branch;
       if a_i.TYPE == OR-join
           submit(a_i, right-end(Domain(a_i));
       until end-of-graph
end
```

Fig. 5. Strict-True-Agent Pseudo-Static Scheduler:

Example: *In order to demonstrate the need for taking the worst-case solution we slightly alter the emergency room example presented in the introduction. Activity a_{11} and a_{12} have duration 1.*

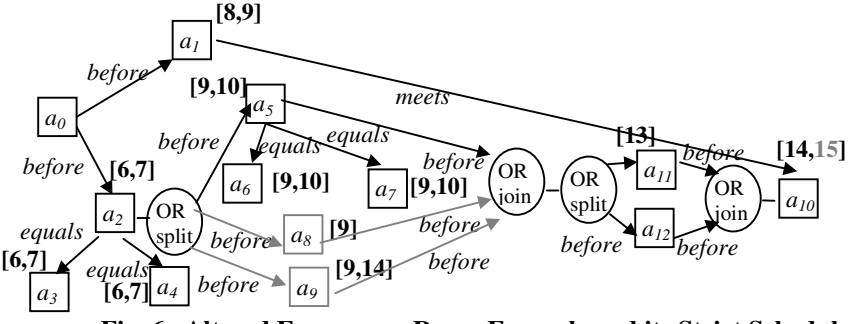


Fig. 6. Altered Emergency Room Example and its Strict Schedule

In Figure 6, the worst-case execution when the first OR branches are to be followed is: (a_0,0), (a_1,9), (a_2,7), (a_3,7), (a_4,7), (a_5,10), (a_6,10), (a_7,10), (a_{11},13) and (a_{10},15). The working of the scheduler gives: 1st iteration: (a_0,0),(a_1,9),(a_2,7), (a_3,7), (a_4,7),are submitted until the OR-split construct is reached. At that point the scheduler waits until the activities are executed and the branch to be followed is determined, 2nd iteration:(a_5,10), (a_6,10),(a_7,10) where the next OR-split construct is reached,3rd iteration: (a_{11},13), 4th iteration: (a_{10},15) where the end of graph node is reached. The workflow instances complete at time 17. Since the first branch was selected, we could have scheduled a_1 at time 7 and complete the workflow at time 15. However, the earliest time we can take the decision about the start time of activity a_1 is after we have decided the domain of a_{10}, which is at time 13. However, then it is too

late to schedule a_i at time 7. Therefore, we schedule assuming the worst-case branch was taken and use the right-ends of the domains.

b)soft-agent pseudo-static scheduler: When dealing with soft correctness we can tolerate deviation from the specified execution. The soft-agent pseudo static scheduler executes at run-time the DDCA algorithm to decide upon the path to be followed. The advantage is that an agent can finish the activity execution earlier than the hard true agent can but we allow inconsistencies sometimes. The algorithm iterates dynamically between OR constructs computing the domains of activities. Those activities that have all their constraints evaluated are submitted for execution. For the rest, the scheduler checks whether their right end of the domain has been reached, if so, it submits them.

```
Soft-True-Agent Pseudo-Static Scheduler (SoTAPS):
Input: Time-Constrained Process Graph (TG)
SoTAPS(graph: TG)
begin
   repeat
      ai=next_node(TG);
      if ai.TYPE == activity insert(ai,Q);
      if ai.TYPE == OR-split evaluate branch;
      if ai.TYPE == OR-join
         DDCA(Q);
         ∀ai∈Qif all constraints are visited ai.VISITED= TRUE;
         ∀ai∈Q|VISITED==TRUE submit(ai, left-end(Domain(ai));
         cn=sn+durn|an is last in Q,sn=left-end(Domain(an))
         ∀aj∈Q | VISITED==FALSE  if right-end(Domain(aj)) < cn
            submit(ai, right-end(Domain(ai));
         check_constraints( );  //checks consistency
   until end-of graph
end
```

Fig. 7. : Soft-True-Agent Pseudo-Static Scheduler

Upon reaching an OR-join construct it evaluates whether an inconsistency occurred. If so, it starts the exception handling mechanism or aborts the instance. We are optimistic that the activities start their execution on time, otherwise we achieve at least the deadline compliance. There are no dependencies (other than *before*) among activities that span different OR-constructs. Such a dependency would imply that the specific branch should be taken always and there is no need for an OR-construct.

c) Agent pseudo-static scheduling Agents are not always able to follow global scheduler directions. They might delay the start or completion time of activities. We assume that agents can return s_i'/c_i', which is the time activity a_i was started/completed. In this case, the global scheduler still tries to achieve the earliest possible execution. When an agent returns a not-true value (that is, the activity was not started or completed at the time scheduled), then the global scheduler has to verify whether this violation creates an inconsistency or there can be still a consistent execution. It does this by re-executing the DDCA algorithm to all activities affected by the different start/completed time returned. If the DDCA can still assign start values to all subsequent activities then workflow instance continues executing (that is, there is no inconsistency), otherwise it reports an inconsistency. In that case, exception handling is initiated.

6. Conclusions

In this work, we proposed a framework (T-WfMc) for incorporating time-management information in workflow execution and specification. We argue that the current approach for handling time-related issues in workflow contexts is inadequate to support sophisticated time control needed by complex workflow applications. Further, we note that the temporal constraints plus deadlines is much more stringent case of temporally constrained workflow systems than the case where only deadlines are used for activity completion times. We provided tools for checking both the consistency of the specification and the consistency of activities' duration and deadline. We showed the architecture needed to support T-WfMc processes. We presented global scheduling algorithms tailored to the specific needs and requirements of the workflow environment and applications (specified by using different parameters). We illustrated by using real-life examples the practicality of the temporal constrained workflow systems that is advocated in this paper. Further, research is needed to evaluate and design algorithms for global and agent scheduling.

References

[A-83] Allen J.F., "Maintaining knowledge about temporal intervals", *Communications of the ACM*, 26:832-843, 1983.

[AH-97] Alonso G., Hagen C.,"Workflow Concepts for Spatial Processes, *SSD'97*.

[AM-97] Alonso G., Mohan C., " WFMS: the next generation of distibuted processing tools", *In Advanced Transaction Models and Architectures*,1997.

[ASSR-93] P.C. Attie, M.P. Singh, A.P. Sheth, and M. Rusinkiewicz, "Specifying and Enforching Intertask Dependencies" *VLDB*, 1993

[C-97] Mohan C. "Tutorial:State of the Art in Workflow Management Systems Research and Products*", DASFAA* '97, Australia.

[DMP-90] Dechter R., Meiri I. and Pearl J., Temporal Constraints Networks, *Artificial Intelligence*, 49, pp 61-95, Elsevier, 1991.

[DR-98] Dadam P., Reichert M., "The ADEPT WfMS project at the university of Ulm",*WPM'98*, Switzarland, 1998

[EPR-99] Eder .J, Panagos E., Rabinovich M., "Time Constraints in Workflow Systems", *CAiSE 99,* Germany.

[H+-96] Haimowitz I., Farley J., Fields G.S., Stillman J and Vivier B., "Temporal Reasoning for automated workflow in Helath Care Enterprises", *Electronic Commerce: Current Reasearch Issues and Applications*, Springer-Verlag,1996.

[HDT-92] Hentenryck P., Deville Y., Teng C., "A generic arc-consistency algorithm and its specializations", Artificial Intelligence vol 57, 291-321, 1992.

[KR-98] Kamath M., Ramamritham K., "Failure Handling and Coordinated execution of Concurrent Workflows", ICDE '98.

[M-77] Mackworth A., Consistency in Networks of Relations, *Artificial Intelligence* 8:99-118, 1977.

[RS-94] Rusinkiewicz M. and Sheth A., "Specification and Execution of Transactional Workflows", The Object Model, Interoperability, and Beyond, Addison-Wesley, 1994.

[Vb-92] Van Beek P. ``Reasoning about qualitative temporal information", *Artificial Intelligence*, 58:297-326,1992.

[WfMc] Workflow Management Coalition, *The Workflow Reference Model*, The Workflow Management Coalition 1995. Accessible via http://www.aiim.org/wfmc.

[ZS-99] Zhao L. J. and Stohr E. A., "Temporal Workflow Management in a Claim Handling System, Int. Joint Conf. on Work Activities Coordination and Collaboration, 1999.

Temporal Modeling of Workflows with Conditional Execution Paths

Johann Eder[1], Wolfgang Gruber[1], and Euthimios Panagos[2]

[1] Department of Informatics-Systems, Univ. Klagenfurt, A-9020 Klagenfurt, Austria
{eder,gruber}@isys.uni-klu.ac.at
[2] AT&T Labs - Research, 180 Park Avenue, Florham Park, NJ 07932
thimios@research.att.com

Abstract. In this paper, we present a novel technique for modeling, checking, and enforcing temporal constraints in workflow processes containing conditionally executed activities. Existing workflow time modeling proposals either do not discriminate between time constraints that apply to disparate execution paths, or they treat every execution path independently. Consequently, superfluous time constraint violations may be detected at modeling time, even when each execution path does not violate any constraints. In addition, scheduling conflicts during process execution may not be detected for activities that are common to multiple execution paths. Our approach addresses these problems by (partially) unfolding the workflow graph associated with a process that contains conditionally executed activities and, then, incorporating the temporal constraints in the time calculations performed on the unfolded graph.

1 Introduction

Today, the most critical need in companies striving to become more competitive is the ability to control the flow of information and work throughout the enterprise in a timely manner. Workflow management systems (WFMSs) improve business processes by automating tasks, getting the right information to the right place for a specific job function, and integrating information in the enterprise [GHS95,Law97,Wor94,Hol95]. However, existing WFMSs [LR94,InC,Flo] offer limited support for modeling and managing time constraints associated with processes and their activities [PEL97]. This support appears mainly in the form of monitoring activity deadlines [Sch96]. However, the consistency of these deadlines and the side effects of missing some of them are not addressed.

In process centered organizations, time management is essential for process modeling and management. Many business processes have restrictions such as limited duration of subprocesses, terms of delivery, dates of re-submission, or activity deadlines. Typically, time violations increase the cost of a business process because they lead to some form of exception handling [PR97b]. Therefore, a WFMS should provide the necessary information about a process, its time restrictions, and its actual time requirements to a process manager. In addition, the process manager needs tools to anticipate time problems, proactively avoid

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 243–253, 2000.

time constraints violations, and make decisions about the relative priorities of processes and timing constraints.

The notion of *timed workflow graphs* was introduced in [EPPR99], and it was shown how the time information represented in these graphs can be used at modeling, process instantiation, and execution times to manage workflow executions without time errors. In [EPR99], we introduced *explicit temporal constraints* and presented a technique for incorporating these constraints into timed workflow graphs. However, the technique presented in [EPR99] treated temporal constraints associated with parallel and conditionally executed activities in the same way. In this paper, we show that differentiating between these cases is crucial in avoiding many superfluous constraint violations, and we present the (partial) unfolding technique for handling them.

2 Workflows, Constraints, and Timed Graphs

In this section, we present the necessary definitions, assumptions, and methods used in the remainder of the paper.

2.1 Workflows

A workflow is a collection of *activities, agents,* and *dependencies* between activities. Activities correspond to individual steps in a business process, agents (software systems or humans) are responsible for the enactment of activities, and dependencies determine the execution sequence of activities and the data flow between them. We assume that workflows are *well structured*. A well-structured workflow consists of m sequential activities, $T_1 \ldots T_m$. Each activity T_i is either primitive, i.e., it cannot be decomposed any further, or composite. A composite activity consists of n_i parallel conditional or unconditional sub-activities ${T_i}^1, \ldots, {T_i}^{n_i}$, each of which is either primitive or composite. Typically, well structured workflows are generated by workflow languages that provide the usual control structures and adhere to a structured programming style of workflow definitions, such as Panta Rhei [EGL97].

Workflows are represented by *workflow graphs*, where nodes represent activities and edges correspond to dependencies between activities. An *and-split* node refers to an activity having several immediate successors, all of which are executed in parallel. An *and-join* node refers to an activity that is executed after all of its immediate predecessors finish execution. An *or-split* node refers to an activity whose immediate successor is determined by evaluating some boolean expression. An *or-join* node refers to an activity that joins all the branches after an or-split. Finally, similar to [MO99], a *workflow instance type* refers to workflow instances that contain exactly the same activities, i.e., for each or-split node in the workflow graph, the same successor node is chosen.

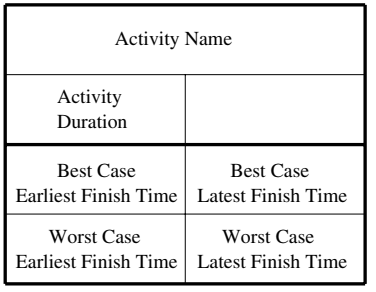


Fig. 1. Activity node of a timed workflow graph

2.2 Temporal Constraints

Time is expressed in some basic time units relative to a time origin, which is usually the start of the workflow. In addition, each activity has a duration which, for simplicity, is assumed to be deterministic. Activity durations and control dependencies between activities determine the *structural time constraints*. These constraints arise from the fact that an activity can only start when its predecessor activities are finished. In addition, workflow designers may specify *explicit time constraints*, i.e., temporal relations between start and end of (different) activities. These constraints are derived from organizational rules, laws, commitments, and so on (e.g., an appeal can be filed within 7 days after the verdict, a meeting invitation has to be sent to all participants at least one week before the meeting). In particular, the following explicit time constraints can be specified.

- **Lower bound constraint ($lbc(s, d, \delta)$):** The time distance between source event s and destination event d must be greater than or equal to δ.
- **Upper bound constraint ($ubc(s, d, \delta)$):** The time distance between source event s and destination event d must be smaller than or equal to δ.

2.3 Timed Workflow Graphs

Our time constraint management techniques are based on the notion of a *timed workflow graph*, which extends the workflow graph by augmenting each activity node n with the following[1]. (Figure 1 shows the representation of such a node.)

- $n.E^{bc}$: The earliest point in time n can finish when the shortest path is chosen to reach n;
- $n.E^{wc}$: The earliest point in time n can finish when the longest path is chosen to reach n;
- $n.L^{bc}$: The latest point in time n has to finish in order to meet the overall deadline via the shortest path;

[1] Since activity durations are assumed to be deterministic, start times for activities are computed by subtracting their durations from their termination times.

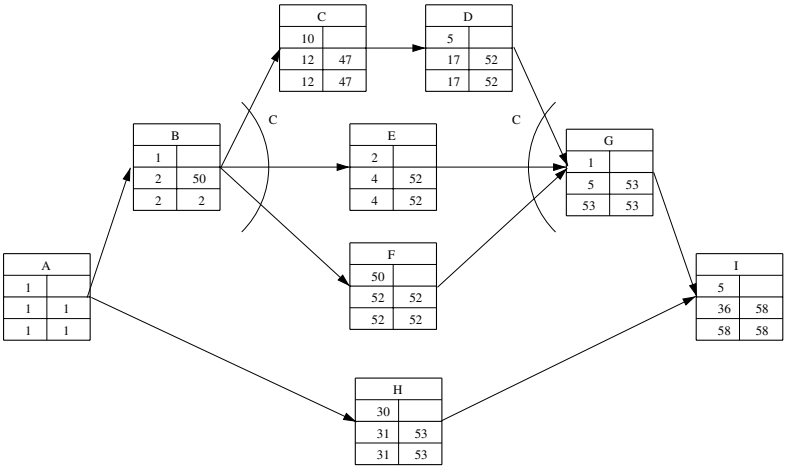


Fig. 2. Example timed workflow graph

– $n.L^{wc}$: The latest point in time n has to finish in order to meet the overall deadline via the longest path.

Without explicit time constraints, the above values can be computed by extending the Critical Path Method (CPM) [Phi86] to handle conditional execution paths [PEL97]. Table 1 shows the actual computations, where $d(n)$ denotes the execution duration of activity n. E-values are computed in a forward pass, with the E-values of the starting workflow activity being set to its duration. L-values are computed in a backward pass, with the L-values of the last workflow activity equal to its E-values.

Figure 2 shows the timed workflow graph we use in the rest of the paper. In this graph, activity A is followed by an and-split, having I as the and-join, and activity B is followed by an or-split, having G as the or-join. The values of node I show the duration of the workflow, i.e., $I.E^{bc}$ and $I.E^{wc}$ indicate that

FORWARD	**best case (bc)**	**worst case (wc)**
sequence	$E_j = E_\tau + d(j)$	$E_j = E_\tau + d(j)$
and-join	$E_j = \max(\{E_\tau + d(j)\})$	$E_j = \max(\{E_\tau + d(j)\})$
or-join	$E_j = \min(\{E_\tau + d(j)\})$	$E_j = \max(\{E_\tau + d(j)\})$
	$\forall$ immediate predecessor τ of j	
REVERSE	**best case (bc)**	**worst case (wc)**
sequence	$L_j = L_\tau - d(\tau)$	$L_j = L_\tau - d(\tau)$
and-split	$L_j = \min(\{L_\tau - d(\tau)\})$	$L_j = \min(\{L_\tau - d(\tau)\})$
or-split	$L_j = \max(\{L_\tau - d(\tau)\})$	$L_j = \min(\{L_\tau - d(\tau)\})$
	$\forall$ immediate successor τ of j	

Table 1. Calculation instructions for timed workflow graphs

the workflow execution may take between 36 and 58 time units. $G.E^{bc}$ indicates that G may finish after 5 time units (when E follows B), while $G.E^{wc}$ indicates that no path from A to G should take more than 53 time units. $B.L^{bc}$, tells us that if B is finished at time point 50, the workflow may still be able to terminate in time. From $B.L^{wc}$ we learn that if B is finished at time point 2, we can meet the overall deadline, irrespective of the conditionals.

3 Incorporating Explicit Time Constraints

Once the timed workflow graph is constructed, we can incorporate explicit time constraints into it by using the algorithms shown in [EPR99]. Lower bound constraints are incorporated during the construction of the timed workflow graph; they may increase E-values during the forward pass and decrease L-values during the reverse pass. On the other hand, the incorporation of upper-bound constraints should check for constraint violations; for $ubc(s, d, \delta)$, $s.E + \delta < d.E$ and $s.L + \delta < d.L$. When a constraint is violated, the E- and L-values of s and d are shifted in an attempt to satisfy the constraint, with the invariant that an E-value is not greater than its corresponding L-value.

During the incorporation of explicit constraints the semantics of the E-values change: the E-values mandate that activity terminations should not occur earlier than them in order to meet the time constraints. Therefore, the E- and L-values define the time interval during which an activity has to terminate. This time interval is referred to as the *life-line* of the activity.

However, [EPR99] does not discriminate between conditional and unconditional branches in the computation of worst case E- and L-values. While this ensures that the execution of the workflow will avoid violating temporal constraints when the incorporation algorithm succeeds, it is overly pessimistic. In particular, there are cases where execution without constraint violation is possible and the incorporation algorithm does not succeed due to interference of constraints on mutually exclusive conditional branches, as we show below.

In general, the following issues need to be addressed when we derive timed graphs that violate explicit time constraints.

1. *Checking individual constraints for violation may not be sufficient.* As shown in [EPR99], a set of time constraints may not be satisfiable, even when each individual constraint is satisfiable. Consequently, the incorporation procedure should consider all constraints together.
2. *Checking workflow instance types for constraint violation in isolation is not sufficient.* If two instance types only differ after or-splits, their common initial activities should have the same E- and L-values. If we cannot find such E- and L-values in all instance types to satisfy the constraints, then it may not be possible to schedule the execution of this workflow so that all time constraints are met. For example, consider two instance types for the workflow in Figure 2, where one includes C and D and results in $B.E^{wc} = 20$ and $B.L^{wc} = 40$, and the other one includes E and results in $B.E^{wc} = 5$ and $B.L^{wc} = 15$. Here, it is not possible to satisfy the time constraints for

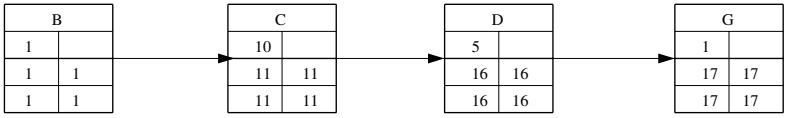


Fig. 3. Path with $ubc(B, D, 20)$ and $ubc(C, G, 15)$

either instance since B is executed in both, and the scheduling information for B, i.e., valid interval for B to terminate, is contradictory.

3. *Incorporating upper-bound constraints using best-case values may not be meaningful.* This can be seen by looking at the best-case values of C and G in Figure 2. The values of C do not really contribute to the values of G because they are dominated by those of E and F. Therefore, checking an upper-bound constraint between C and G for the best-case is not possible.
4. *Checking violation of upper-bound constraints using worst-case values may lead to unnecessary rejections when the workflow has conditional branches.* This can be seen by examining the case where $ubc(B, D, 20)$ and $ubc(C, G, 15)$ need to be incorporated into the timed graph shown in Figure 2. If we incorporate $ubc(B, D, 20)$ first, then $D.L^{wc}$ becomes 22 and, consequently, $ubc(C, G, 15)$ cannot be satisfied. However, each of these constraints is individually satisfiable, as shown in Figure 3, since the path containing C and D does not influence the computation of the worst case E- and L-values of nodes B and G.

4 Unfolded Workflow Graph

To address the time constraint incorporation problems, we construct the *unfolded timed workflow graph*. This graph contains exactly the same set of instance types as the original graph. However, it does not contain or-joins and has several termination nodes, one for each instance type. Once a workflow graph is unfolded, we are able to assign different time information to activities in disparate instance types after their separating split node, and share the same time information for activities before this node. In this way, explicit temporal constraints that involve activities in different instance types no longer interfere and, thus, we may be able to satisfy all of them.

Another advantage of unfolding a workflow graph is that different deadlines, i.e., worst case E- and L-values, may be assigned to activities belonging to different instance types. Consequently, different deadlines for the final activities of some instance types may be computed. As shown in [PR97a], combining the different deadlines for disparate instance types with the probabilities of executing such instance types is beneficial.

Intuitively, the unfolded timed workflow graph is derived from the original workflow graph by duplicating the graph at or-joins. Therefore, two instance types share the same nodes before the very first or-split that distinguishes them

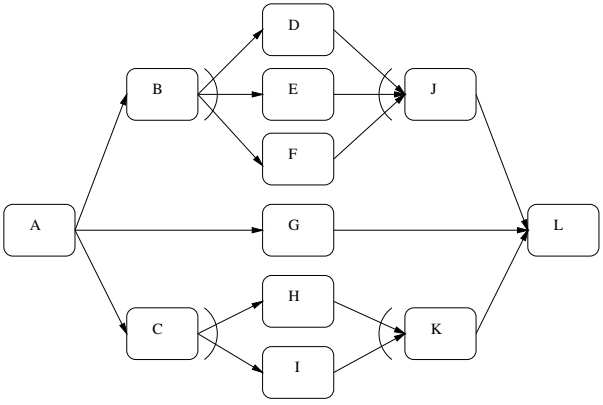


Fig. 4. Aggregated workflow graph

and have different nodes for activities thereafter, even for activities shared after the or-join that merges these instances in the original graph. In practice, however, the unfolding procedure is more complicated since all unconditionally executed parallel branches should be closed with and-joins. We explain how this is done below.

4.1 Unfolding Procedure

The procedure for generating an equivalent unfolded workflow graph U for a workflow graph G is as follows. First, the start node of G is copied into U. Then all nodes of G are visited in topological order. Copies of each node n of G, which is not an and-join, are inserted into U as many times as there are copies of the predecessors of n and the nodes are connected accordingly, such that each copy of node n is connected with exactly one copy of a predecessor of n and vice versa[2].

If n is an and-join node, then we place a copy of n in U for all valid predecessor combinations. These combinations are computed by constructing all combinations of copies of predecessor nodes of n in G, such that in each combination there is exactly one copy of each of these predecessor nodes (i.e., the Cartesian product of the copies of the respective nodes). Figure 5 shows the unfolded graph for the workflow graph shown in Figure 4.

After the construction of the unfolded workflow graph, the temporal constraints are mapped into this workflow graph. The mapping is done in such a way that if a constraint with source s and destination d exists in the original graph G, then this constraint exists between all copies of s and d that belong to the same instance type in the unfolded graph U. Once we are done mapping the constraints, we can compute the timed workflow graph based on the unfolded

[2] Copies of G's or-join nodes have exactly one predecessor node in U and, thus, there are not or-joins anymore.

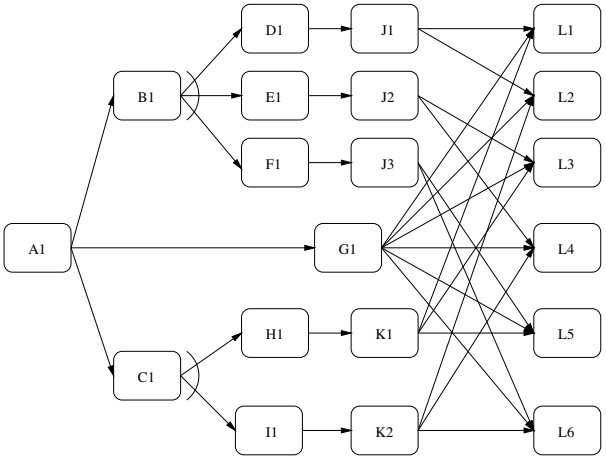


Fig. 5. Unfolded workflow graph

graph and then incorporate the temporal constraints using a variation of the algorithms presented in [EPR99].

While the above procedure addresses the constraint incorporation problems of Section 3, it suffers from the potential explosion of the number of "duplicate" nodes in the unfolded graph, since it considers each instance type separately. This is not desirable when discriminating between instance types is not necessary because either there are no interfering constraints in these instance types or we can check the satisfiability of such constraints without unfolding. To address this problem, we developed the *partial unfolding* technique.

4.2 Partial Unfolding of Workflow Graphs

Since constraints need not appear in every alternative path, we can unfold the workflow graph only where it is necessary to check constraints. We call such partially unfolded graphs "hybrid graphs". The procedure for partially unfolding a workflow graph G to a hybrid graph H begins by selecting a *hot-node*, with the side effect that all instance types going through the hot-node are factored out, or intuitively, the workflow graph reachable from the hot-node is duplicated. In principle, every node can be hot-node. For practical reasons, we require that a hot-node is an immediate predecessor of an or-join. In the next section we will show how hot-nodes are chosen, when a time constraint cannot be incorporated. Once a hot-node is identified, partial unfolding takes place as follows:

1. Copy the original graph G into H;
2. Insert an additional copy of all nodes reachable from the hot-node, together with a copy of all edges between such nodes;
3. Remove the edge from the copy of the hot-node to it's original successor and include an edge to the created copy of the successor node;

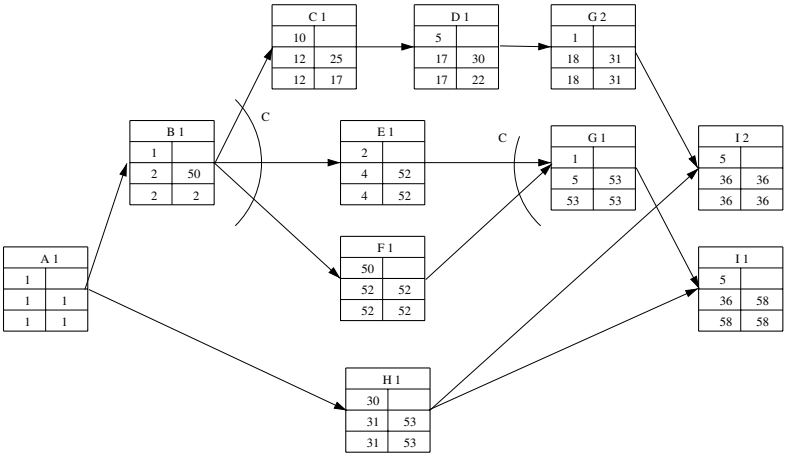


Fig. 6. Partially unfolded graph with $ubc(B1, D1, 20)$ and $ubc(C1, G2, 15)$

4. For all edges from a node n in G, which is not successor of the hot-node, to an and-join node a in the succession of the hot-node, include a copy in H;

Based on this procedure, we only have to partially unfold the workflow graph shown in Figure 2 at node D in order to check the satisfiability of $ubc(B1, D1, 20)$ and $ubc(C1, G2, 15)$. Figure 6 shows the resulting partially unfolded graph.

4.3 Computation of the Timed Graph

We first compute the timed graph using structural constraints and any explicit lower-bound constraints. Then, we attempt to incorporate all upper-bound constraints. When an upper-bound constraint is violated, we determine whether its source and destination nodes are connected via conditionally executed activities or they belong to the same workflow instance type. Here, we first determine the hot-nodes, then we partially unfold the workflow graph and, finally, we attempt the constraint incorporation procedure again.

For checking the satisfiability of time constraints, we extended the algorithm in [EPR99] to the full nodes (best/worst case). If a constraint $ubc(s, d, \delta)$ cannot be incorporated, the algorithm terminates with an error. In this case, we determine a node s' as the first successor node of s that immediately precedes an or-join and all paths from s to s' have the same number of or-joins and or-splits[3]. In the same way, we determine node d' for destination node d. If such nodes exist, we restart the incorporation procedure on the partially unfolded workflow graph that used these nodes as hot-nodes. Otherwise, either the path has been unfolded already, or the nodes do not have multiple copies in the unfolded graph.

[3] Intuitively, we search for the or-join reached from s that joins the conditional branch through s with its parallel branches and take the immediate predecessor of this or-join as s'.

If a constraint is violated and its source and destination nodes cannot be used for unfolding, then we check whether there is an overlapping constraint and perform the unfold for the source and destination nodes of this constraint. An example for this procedure is given in the workflow shown in Figure 2 and the constraints $ubc(B, D, 20)$ and $ubc(C, G, 15)$. If $ubc(B, D, 20)$ is incorporated first, then $ubc(C, G, 15)$ cannot be incorporated since $D.L^{wc}$ is 22. Now we take D as hot-node and partially unfold the workflow graph, computing the timed hybrid graph, and incorporating the constraints there. The result of this procedure is the graph shown in Figure 6.

The algorithms for incorporating explicit time constraints in timed workflow graphs with (partial) unfolding have been implemented in a prototype for an extended workflow design tool. The prototype accepts workflow descriptions in the workflow definition language of the workflow system Panta Rhei [EGL97], extended with explicit time constraints. The algorithms for unfolding and partially unfolding workflow graphs as well as the algorithms for computing timed workflow graphs and incorporating explicit time constraints into these timed graphs are defined on these process definitions.

5 Related Work

Incorporating explicit time constraints into the modeling and management infrastructure of WFMSs has received very little attention from both workflow vendors and researchers. Among the work that is available in the literature, our work is closely related to [EPR99,MO99]. In particular, we extended [EPR99] to explicitly handle temporal constraints associated with conditionally executed activities by unfolding the timed workflow graph of a process. By doing so, we are able to avoid many superfluous constraint violations.

In contrast to [MO99], we do not consider time constraints in isolation and provide solutions for overlapping, interleaving, and interfering constraints. As we demonstrated in [EPR99], a set of time constraints may be unsolvable (i.e., there is no workflow instance that does not violate at least one time constraint) even when every single constraint is solvable in isolation. In addition, our techniques are pro-active in nature, and they attempt to modify the E- and L-values of activities in order to make constraints satisfiable.

6 Conclusions

In this paper, we presented a new technique for modeling, checking, and enforcing temporal constraints in workflow processes containing conditionally executed activities. Our technique discriminates between time constraints that apply to disparate execution paths and, thus, it avoids the superfluous time constraint violations detected by existing techniques that treat these paths similar to those of unconditionally executed activities. In addition, our graph unfolding procedure and the incorporation of explicit time constraints into the unfolded graph avoid

the problem of detecting scheduling conflicts when workflow instances are treated independently of each other.

References

[EGL97] Johann Eder, Herbert Groiss, and Walter Liebhart. The Workflow Management System Panta Rhei. In A. Dogac et al. (eds.), *Advances in Workflow Management Systems and Interoperability*. Springer Verlag, 1997.

[EPPR99] Johann Eder, Euthimios Panagos, Heinz Pozewaunig, and Misha Rabinovich. Time management in workflow systems. In *Business Information Systems BIS'99*, pages 265–280. Springer Verlag, 1999.

[EPR99] Johann Eder, Euthimios Panagos, and Misha Rabinovich. Time constraints in workflow systems. In *Proc. Int. Conf. CAiSE'99*. Springer Verlag, 1999.

[Flo] TeamWare Flow. Collaborative workflow system for the way people work. P.O. Box 780, FIN-00101, Helsinki, Finland.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: From process modeling to workflow automation. In A. Elmagarmid, ed., *Distributed and Parallel Databases*, vol 3. Kluwer, 1995.

[Hol95] D. Hollingsworth. The workflow reference model. Draft 1.1 TC00-1003, Workflow Management Coalition, July 1995.

[InC] InConcert. Technical product overview. XSoft, a division of xerox. 3400 Hillview Avenue, Palo Alto, CA 94304. http://www.xsoft.com.

[Law97] P. Lawrence. *Workflow Handbook 1997*. John Wiley & Sons, 1997.

[LR94] F. Leymann and D. Roller. Business process management with flowmark. In *Proceedings of the 39th IEEE Computer Society International Conference*, pages 230–233, San Francisco, California, February 1994.

[MO99] O. Marjanovic and M. Orlowska. On modeling and verification of temporal constraints in production workflows. *Knowledge and Information Systems*, 1(2), May 1999.

[PEL97] H. Pozewaunig, J. Eder, and W. Liebhart. ePERT: Extending PERT for Workflow Management Systems. In *First EastEuropean Symposium on Advances in Database and Information Systems ADBIS ýý7*, 1997.

[Phi86] Susy Philipose. *Operations Research - A Practical Approach*. Tata McGraw-Hill, New Delhi, New York, 1986.

[PR97a] E. Panagos and M. Rabinovich. Predictive workflow management. In *Proceedings of the 3rd International Workshop on Next Generation Information Technologies and Systems*, Neve Ilan, ISRAEL, June 1997.

[PR97b] E. Panagos and M. Rabinovich. Reducing escalation-related costs in WFMSs. In A. Dogac et al.(eds.), *NATO Advanced Study Institue on Workflow Management Systems and Interoperability*. Springer Verlag, 1997.

[Sch96] G. Schmidt. Scheduling models for workflow management. In B. Scholz-Reiter and E. Stickel, editors, *Business Process Modelling*. Springer, 1996.

[Wor94] Workflow Management Coalition, Brussels, Belgium. *Glossary: A Workflow Management Coalition Specification*, November 1994.

Verified Order-Based Transaction Scheduling Scheme for Multilevel Secure Database Management Systems

Yonglak Sohn [1] and Songchun Moon [2]

[1] Dept. of Computers Engineering, Seokyeong Univ., 16-1 Jungneung-Dong Sungbuk-Ku, Seoul 136-704, Korea, Tel.: +82-2-940-7103, Fax.: +82-2-919-0345
syl@bukak.seokyeong.ac.kr

[2] Database Sys. Lab., Graduate School of Management, Korea Advanced Institute of Science and Technology, 207-43 Cheongryangri-Dong, Dongdaemun-Ku, Seoul 130-012, Korea
moon@cais.kaist.ac.kr

Abstract. While the secure transaction schedulers in multilevel secure database systems synchronize transactions cleared at different security levels, they must consider the problem of *covert channel*. Through the covert channel, malicious users leak secret information in a way of intentional interference among the transactions that they invoked. Much work had been done for closing the covert channel. Although they succeeded in closing the covert channel, they unfortunately failed in preserving correctness, sufficient recentness of versions read, or fairness with respect to availability. In this paper, we present a new secure transaction scheduler, named *Verified Order-based Transaction Scheduler* (VO) that founds on multiversion database. VO overcomes the problems of previous work. *Keywords*: *Multilevel security, Database, Concurrency Control, Covert Channel*

1 Introduction

While transactions execute concurrently and share data objects, conflicts among them are unavoidable. Conventional transaction schedulers preserve correctness in a way of interference such as rejecting or delaying the conflicting operations [7]. Although these conventional ones preserve correctness with regard to the notion of serializability, they are definitely vulnerable to an infringement of *secrecy-focused security*.[1] If such interference occurs from a transaction cleared at high security level, high transaction for short, to low transactions, low transactions for short, it opens an unexpected communication channel, named *covert channel*, between the transactions.

Eliminating the covert channel is one of the major requirements of security evaluation criteria for secure information systems. The problem of covert channel now com

[1] In principle, security in computer system is composed of three major aspects: secrecy, integrity and availability. However, we focus only on secrecy. Therefore, the term *security* in this paper represents the aspect of *secrecy*.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 254-263, 2000.

pulsorily demands the attention to security along with correctness. It therefore makes secure transaction schedulers more complex than conventional transaction schedulers.

Much work, such as [2, 4, 5, 6, 9, 10, 11, 12] have been devoted to develop secure transaction schedulers(STSs). Some of them [2, 5, 9] argue that the traditional notion of serializability-related correctness [7] is too restrictive for MLS/DBMS to achieve desirable performance. On the other hand, STSs proposed by [4, 6, 11] achieve correctness and security at the cost of declined performance of high transactions. They thus come to infringe fairness with respect to availability of transactions of different security levels. STSs proposed by [10, 12], achieve security and correctness in a way of disposing high transactions before active low transactions with respect to serializability. Accordingly, high transactions are now forced to read far outdated data versions.

Verified Order-based transaction scheduler (*VO*) proposed in this paper founds on multiversion database (MVDB) and eliminates covert channel in a way of driving all the abort-destined conflicts to occur only among transactions cleared at the same security level. The major scheduling policies of VO are as follows:

- Non-interfered executions of conflicting operations do not always violate correctness or security.
- Retaining sufficient trace of ordering relationship among transactions is very much useful for perceiving justice or injustice of non-interfered executions of conflicting operations.

2 The Models

A transaction is an abstract unit of concurrent computation that executes with primitives: *read*, *write*, *commit*, and *abort*. Users interact with the database by invoking transactions. Read or write of a transaction, T_i, that accesses a data item, x, is denoted by $r_i[x]$ or $w_i[x]$, respectively. c_i and a_i denote commit and abort of T_i. History, H, is a set of totally ordered operations that have been scheduled by an STS.

In an MVDB, each write operation of a committed transaction on a data item ultimately produces a new version of the data item that is to be added into a list of versions. $w_i[x]$ and $r_i[y]$ of a transaction, T_i, are now mapped to $w_i[x_i]$ and $r_i[y_j]$ in MVDB. Hence, the new version, x_i, never overwrites its previous versions but instead coexists with them. Coexisting data versions, x_i and x_j, are called *sibling versions* and are totally ordered. x_i comes to precede x_j in case that c_i precedes c_j in a history. In this case, ordering relationship between them is represented by $x_i \ll x_j$. Versions turn out to be visible only after their creators come to commit. The principle of *one-copy serializability*, *1SR* for short, among the scheduled transactions is applied for proving the correctness of MVDB-based history [7].

Security level labeling function defined as follows represents the relationships between transactions and(or) data objects at the standpoint of multilevel security.

Definition 1(Security level labeling function): Security level labeling function, L, maps a transaction and a data object to their specific security levels. It is represented

by L: $T \cup D \rightarrow S$ where D is a set of data objects, T is a set of transactions, and S is a hierarchical set of security levels. $L(T_i) > L(d_j)$ denotes that the security level of T_i is higher than that of d_j.□

In order to filter out unauthorized requests of transactions, multilevel secure information systems need to define an access control policy. In this paper, we define the access control policy as follows:

Definition 2(Access control policy): A transaction T_i is allowed to access a data d_j if $L(T_i)$ and $L(d_j)$ obey following two provisions:

- **Read-equal/Read-down**: T_i is allowed to read from d_j if and only if $L(T_i) \geq L(d_j)$.
- **Write-equal**: T_i is allowed to write a new value to d_j if and only if $L(T_i) = L(d_j)$. □

The access control policy is strictly grounded on Restricted Bell-LaPadula model [1]. By applying this, direct information flow from a transaction, T_i, to T_j with $L(T_i) > L(T_j)$ in a way of writing and reading a value can be cut off. Unfortunately, however, it is insufficient to close covert channel that occurs after the access control policy had been applied. In order to preserve correctness and close covert channel at once, an MVDB-based STS must be able to produce *trustedly serializable*, *TSR* for short, histories all the time. It is defined as follows:

Definition 3(Trusted serializability): A history, H, is TSR if and only if it satisfies following two requirements simultaneously:

- **Correctness preservation**: H is 1SR, and
- **Security preservation**: For any pair of transactions, T_i and T_j with $L(T_i) > L(T_j)$, in H, any interference from T_i to T_j has been eradicated. □

3 Verified Order-Based Transaction Scheduler: *VO*

3.1 Verification of Transaction Ordering Relationship

In order to produce trustedly serializable history, VO maintains information that delineates obviously which transactions precede and which other transactions follow a particular transaction. VO verifies such *transaction ordering relationship*s, *TOR*s for short, in pursuance of following rules whenever it receives an operation. If a transaction, T_i, precedes the other transaction, T_i, TOR between T_i and T_i hereafter is depicted by $T_i \rightarrow T_j$.

Rule 1(Verification of transaction ordering relationship):

- **TOR with regard to read-from**: If $w_j[x_j]$ precedes c_j and c_j precedes $r_i[x_j]$ in a history, VO considers $T_j \rightarrow T_i$.
- **TOR with regard to version order**: If $x_i \ll x_j$, VO considers $T_i \rightarrow T_j$.
- **TOR with regard to read-preceding**: If a history has $r_i[x_j]$ and $x_j \ll x_k$, VO considers $T_j \rightarrow T_i$ and at the same time $T_i \rightarrow T_k$. □

In order to remember the verified TORs, VO maintains following two sets of transactions for each transaction.

Definition 4(Set of preceding transactions and set of following transactions): For a transaction, T_i, there are two sets of transactions, $PT(T_i)$ and $FT(T_i)$, that are composed of transactions which precede and which others follow T_i, respectively. □

When a transaction, T_i, begins, VO initializes $PT(T_i)$ and $FT(T_i)$ to ∅. If VO decides to abort T_i, it removes $PT(T_i)$ and $FT(T_i)$. If T_i precedes T_j, VO sets $PT(T_j)$ to $PT(T_j) \cup \{T_i\} \cup PT(T_i)$. Now that $T_i \in PT(T_j)$, T_j comes to follow T_i and thus all the transactions in $FT(T_j)$ are turned out to follow T_i. Accordingly, VO sets $FT(T_i)$ to $FT(T_i) \cup \{T_j\} \cup FT(T_j)$.

While maintaining the sets, VO adheres following rule.

Rule 2(Non-intersection of preceding and following transactions sets): For a transaction, T_i, VO maintains $PT(T_i) \cap FT(T_i)$ to be ∅ all the time. □

By virtue of such an adherence of non-intersection, a theorem with regard to 1SR is presented as follows:

Theorem 1(Preservation of one-copy serializability with non-intersection of preceding and following transactions sets): A history H is 1SR if and only if $PT(T_i) \cap FT(T_i)$ is ∅ for any transaction, T_i, in H.

Proof: For an arbitrary set of three transactions, T_i, T_j, and T_k in a history, H, let us suppose that $T_j \in PT(T_i)$ and $T_k \in FT(T_i)$. As long as $PT(T_i) \cap FT(T_i)$ is ∅, T_k never precedes T_j and at the same time T_j never follows T_k. Thus, TORs among them are as follows: $T_j \rightarrow T_i$, $T_i \rightarrow T_k$, and $T_k \rightarrow T_j$ never occurs. Accordingly, a partial history that is composed of T_i, T_j, and T_k is now 1SR.

As $T_j \rightarrow T_i$, all the transactions that precede T_j precede T_i. As $T_i \rightarrow T_k$, moreover, all the transactions that follow T_k come to follow T_i as well. Accordingly, $PT(T_i) \cup FT(T_i) \cup \{T_i\}$ composes an extended partial history that covers all the transactions in H to which T_i is relevant. As $PT(T_i) \cap FT(T_i)$ is ∅ still, the extended partial history is now allowed to be 1SR. For any other arbitrary transaction, T_l, if $PT(T_l) \cap FT(T_l)$ is ∅, $PT(T_l) \cup FT(T_l) \cup \{T_l\}$ is 1SR as well. As long as Rule2 is applied successfully for every transaction in H, every corresponding partial history in H is now 1SR. Consequently, H is 1SR. □

3.2 Utilization of Verified Transaction Ordering Relationship

3.2.1 Selection of Versions

For serving a read on multiversion database, providing the latest data version is most desirable. However, simple-minded adherence to the provision of the last versions could lead a history to violate TSR. Accordingly, mapping the read to a second-best version becomes inevitable once in a while. Fig. 1 illustrates such inevitability in a situation of scheduling a read. In Fig. 1, time goes from left to right and thus the fact that op_i positions on the left side of op_j says that op_i precedes op_j.

In Fig. 1, mapping $r_i[x]$ to $r_i[x_j]$ generates a TOR circle, $T_i \rightarrow T_k \rightarrow T_j \rightarrow T_i$: $y_l \ll y_k$ and $r_j[y_l]$ generate $T_i \rightarrow T_k$, $r_j[y_k]$ generates $T_k \rightarrow T_j$ and $r_i[x_j]$ generates $T_j \rightarrow T_i$. According to Theorem 1, this circle violates 1SR. Fortunately, we can preclude the TOR circle by mapping $r_i[x]$ to a second-best version, x_l, instead of x_j in that $x_l \ll x_j$. Through $r_i[x_l]$,

VO disposes T_i prior to T_j. It now substitutes $T_i \rightarrow T_j$ for $T_j \rightarrow T_i$ in the TOR cycle. Now that $T_i \rightarrow T_k \rightarrow T_j$ and $T_i \rightarrow T_j$ are established instead of $T_i \rightarrow T_k \rightarrow T_j \rightarrow T_i$, a TOR cycle among T_i, T_j and T_k has been precluded.

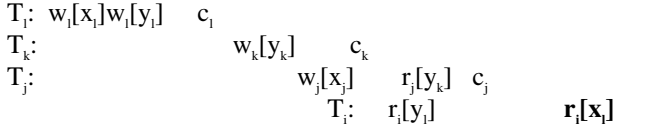


Fig. 1. Preclusion of transaction ordering relationship cycle

Although VO preserves TSR at the cost of mitigating the requests of reading the latest versions, it must be capable of serving a transaction to read a version that is the latest, i.e., the most recent, one among the candidates for being selected. VO selects such an appropriate version in a way of obeying following two rules strictly.

Rule 3 (Trusted version selection):

- **One-copy serializability preserving version selection**: When VO receives a read, it searches for an appropriate data version until it can convince that providing the selected version never produces a TOR cycle.
- **Security preserving version selection**: Suppose that VO is to select x_j as the appropriate version for $r_i[x]$ in accordance with the rule of *1SR preserving version selection*. At this time, if $PT(T_j)$ has an active transaction, T_k, with $L(T_i) > L(T_k)$, VO relinquishes x_j. Instead, it selects the other version, x_l with $x_l \ll x_j$ and $T_k \in PT(T_l)$. In this case, x_l is naturally the latest committed version among its committed siblings that precede x_j. Naturally, 1SR must be preserved after x_l was read. □

In case of *security preserving version selection*, VO relinquishes T_i's reading x_j since reading x_j is highly apprehensive of opening a covert channel between T_i and T_k. If x_j is selected for the appropriate version to $r_i[x]$, T_k comes to precede T_i since $T_k \in PT(T_j)$. If a new TOR, $T_i \rightarrow T_k$, is generated due to the fact that $r_i[z_l]$ precedes $w_k[z_k]$ or $w_k[z_k]$ precedes $r_i[z_l]$ with $z_l \ll z_k$, the TOR between T_i and T_k turns out to be a cycle among T_i and T_k. Since VO cannot assure that it will never receive such a sequence of TOR cycle provoking operations, it is obliged to cope with such a situation prudently in a way of providing a second-best data version to read in advance. By selecting x_l instead of x_j for the appropriate version to $r_i[x]$, VO becomes capable of avoiding $T_k \rightarrow T_i$. After $r_i[x_l]$, although any writes of T_k may establish $T_i \rightarrow T_k$, they never induce a TOR cycle. As a result, such a write is allowed to be processed without being interfered by T_i. Accordingly, VO is now able to close any possible covert channels between T_i and T_k. By obeying rules of *trusted version selection*, consequently, VO is assured to produce TSR preserving histories for these situations.

Note that, however, notwithstanding the application of trusted version selection, when VO can no longer preserve TSR, it unhesitatingly decides to abort a transaction that has requested TSR violating operation. However, such an abort never opens a covert channel. The rationale of this preservation of security is that all the transactions

involved in the situation of TSR violation have been cleared at the same security level. [13] proved this in detail.

3.2.2 Elimination of Redundancies

Restricted accessibility due to given access control policy and maintenance of information about verified TORs enable VO to weed out redundant versions and TORs. This will be of great help to VO for lightening the burdens of maintaining multiple versions and tracing TORs. VO eliminates redundant versions by deciding whether the versions will never be read any more or not whenever it receives a commit. For this, VO applies following rules:

Rule 4(Decision of selective redundant versions): Suppose that there are two adjacent siblings, x_i and x_j with $x_i \ll x_j$, and two sets of active transactions, ST_a and ST_b. In this situation, VO decides that x_i is redundant version and x_j is requisite version if $ST_a \cup ST_b$ is composed of all the active transactions and one of following requirements is satisfied:

- **Readably shading**: x_i and x_j are readable to all transactions in $ST_a \cup ST_b$, or
- **Unreadably shading**: x_i and x_j are unreadable to all transactions in $ST_a \cup ST_b$, or
- **Complexly shading**: x_i and x_j are readable to all transactions in ST_a and at the same time they are unreadable to all transactions in ST_b or vice versa. Note that x_i is readable to T_k if mapping $r_k[x]$ to $r_k[x_i]$ does not violate TSR. □

When a new version, x_i, is created, all the previous siblings of x_i are sometimes turned out to be redundant ones. In this case, fortunately, maintaining only x_i appears to be sufficient for x. VO achieves such a drastic elimination of versions by applying following rule:

Rule 5 (Decision of multiple redundant versions): When a transaction, T_i, commits, VO decides that x_i is a unique requisite version and all the siblings of x_i are redundant ones if one of following conditions meets.

- **Absence of active high or equal transactions**: For every active transaction, T_j, $L(T_i) > L(T_j)$, or
- **Absence of active preceding transactions**: All the transactions in $PT(T_i)$ have already been committed. □

In order to eliminate redundant TORs, on the other hand, VO has a philosophy that all TORs concerned with a transaction whose preceding and following sets of transactions will never intersect are no longer needed to be remembered. For the purpose of definite decision of redundant TORs, VO now applies a rule presented as follows:

Rule 6(Decision of redundant transaction ordering relationship): When VO receives a commit from a transaction, T_i, if there are no active transactions in $PT(T_j)$ of a committed transaction, T_j, that precedes T_i, all TORs concerned with T_j are decided to be redundant ones. □

4 Performance Evaluation

We evaluate the performance of three STSs by means of simulation approach. The simulation is implemented with CSIM [3] discrete event simulation language and much of its model has been borrowed from [8].

4.1 Targets of Comparison

We have excluded STSs that may infringe correctness from comparison since such STSs are liable to degrade integrity. Note that integrity is one of the major aspects of computer security. Among the STSs that are capable of preserving the correctness strictly, we choose *Orange-Locking* method [4] and *Order-Stamping* method [10, 12], hereafter *OL* and *OS* for short, to be compared with VO in that they are representative STSs in the environments of single version and multiversion databases, respectively.

OL schedules transactions of the same security level under the control of conventional two phase locking scheme. When a write of low transaction conflicts with a read of high transaction, OL changes read-lock of the high transaction to *orange-lock* that implies the possibility of incorrect read. When the high transaction commits, OL investigates the serializability. If it detects a violation of serializability, it decides to abort the high transaction unilaterally. Accordingly, the possibility of aborts of high transactions comes to be higher than that of low transactions. When OS schedules transactions of the same security level, on the other hand, it applies conventional multiversion timestamp ordering scheme. For the active transactions of different security levels, OS maintains *order-stamp*s grounded on their security levels. The policy enforces order-stamps of active high transactions to be smaller than those of active low transactions. OS decides the ordering relationships among transactions through the order-stamps and thus high transactions come to precede low transactions in respect of serializability. Accordingly, high transactions are forced to read outdated versions. Moreover, OS leaves alone the redundant versions. This enforces the database to be composed of unbounded number of versions.

4.2 Parameter Settings

The computing resource environment is limited with 2 CPUs and 4 disks. The number of data items in a database is set to 1000 and mean number of operations in a transaction is set to 10 in order to observe interesting performance effects without requiring impossibly long simulation time [8]. The percentage of write operations is set to 10%, 20%, and 30% for observing sensitivities on conflicts. Security levels are ranged from 1 to 12. The number of concurrent transactions, named multiprogramming level, is set to 10, 20 through 200 in steps of 20. 2000 committed transactions are observed after discarding first 400 commits for each run and 10 replications have been applied. All the statistical data presented now have a confidence level of 95% and absolute error of 2%.

4.3 Simulation Results and their Interpretations

The effects of multiprogramming level and percentage of write operations are presented in Fig. 2 and Fig. 3. VO_10 in the figures indicates the average results with write percentage of 10% and scheduled by VO. Abort ratio presented in Fig. 2 means the ratio of number of aborted transactions to total number of transactions submitted. As illustrated, the abort ratio of VO is lower than those of OS's and OL's for the most

part of multiprogramming levels and write percentages. The reason is that VO never aborts a transaction until it meets a situation that non-interfered execution of the selected transaction definitely violates 1SR.

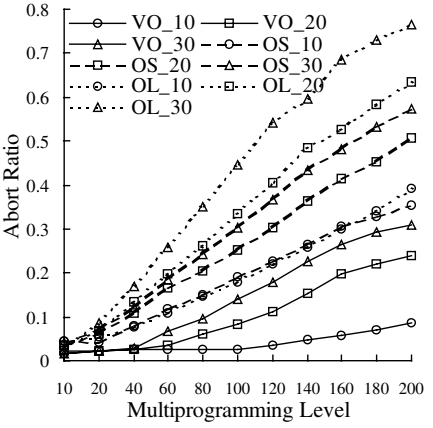


Fig. 2. Abort ratio
(number of security levels = 1,2,3,8,12)

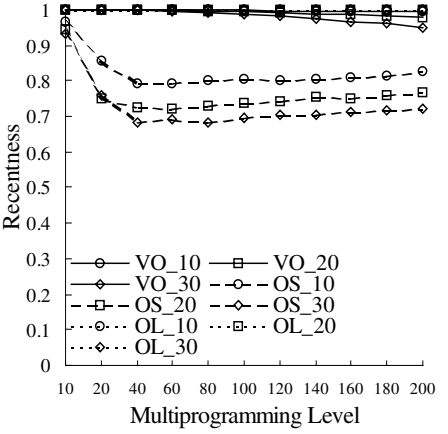


Fig. 3. Recentness
(number of security levels = 1,2,3,8,12)

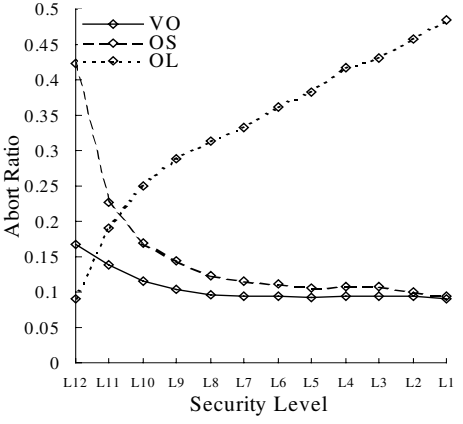


Fig. 4. Abort ratio
(write percentage = 10%,20%,30%,
multiprogramming level = 10,20 through 200
in steps of 20)

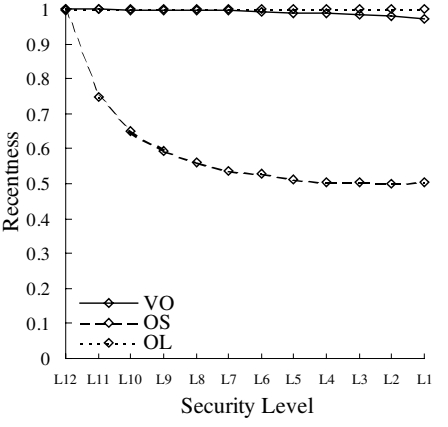


Fig. 5. Recentness
(write percentage = 10%,20%,30%,
multiprogramming level = 10,20 through 200
in steps of 20)

Fig. 3 illustrates that recentness becomes poor as write percentage increases. Recentness of 1 means reading the latest version. The rationale behind this declination in recentness is that the increased write percentage draws up the possibility of TSR violation. In case of VO, however, such probability-based synchronization is restricted to the situation of detecting high potential of security infringement. VO is now proved to provide almost the latest versions.

Fig. 4 and Fig. 5 illustrate the effects of twelve different security levels on the performance of STSs. In the figures, L12 is the lowest and L1 is the highest security level. The abort ratios of VO and OS decrease as security levels go high in that low transactions are destined to conflict each other intensively. Owing to the lack of elaborate information about TORs, the conflicts of OS lead to by far more intensive aborts than those of VO. Transactions cleared at L12 and scheduled by OS now suffer from especially serious aborts. In case of OL, on the contrary, the abort ratio increases as the security levels of transactions go high. The reason is that read-locks of high transactions become more liable to be converted orange-locks as security levels of the transactions go high. Such increment of orange-locks naturally leads to the increment of abort ratio.

The recentness of VO becomes poor as the security levels of transactions go high by applying the rule of security preserving version selection. However, such a declination in the recentness of versions provided by VO is very slight by virtue of referencing TORs that have been maintained elaborately. In case of OS, on the contrary, as transactions are unilaterally forced to be positioned at the ahead of serializable history whenever there are any possibility of TSR violation, almost all of the transactions are obliged to miss a chance of reading the latest versions. On the other hand, reading versions that are outdated excessively induces a problem of accumulating redundant versions in a database. The reason is that OS does not have any information from which it can convince that the versions will never be read any more. In case of VO, by virtue of eliminating redundant versions, only 2.4 siblings of a data item in average are required to be maintained in a database [13].

5 Conclusions

We have proposed a new secure transaction scheduler, named Verified Ordering-based secure transaction scheduler (VO), that founds on multiversion database. It utilizes information, which elaborately describes ordering relationships among transactions(TORs), for the purpose of verifying whether the newly arrived operation definitely violates one-copy serializability or has a high potential of opening a covert channel. By referencing the information, VO becomes capable of guarding transactions against being aborted unnecessarily and reading excessively outdated data versions. By virtue of the elaborate information about transaction ordering relationships, in addition, VO is capable of deleting redundant versions and redundant transaction ordering relationships. The ability for perceiving such redundancies has the advantage of reducing the intrinsic overhead for maintaining multiple versions and accumulated transaction ordering relationships. The profit gained by maintenance of transaction ordering relationships has been proved through the performance evaluation.

In principle, computer security is composed of three major aspects: secrecy, integrity, and availability. In this paper, we have mainly focused on achieving secrecy. Integrity and availability have been achieved without precise modeling from the viewpoint of security. We therefore hope to study the issues for modeling integrity and availability in multilevel secure information systems. Founded on the studies, we expect to develop a secure transaction scheduler whose capability of security-guarding has been strengthened by far.

References

[1] D. E. Bell and L. J. LaPaduda, Secure Computer Systems: Unified Exposition and Multics Interpretation, Tech. Rep. MTR-2997, The Mitre Corp., 1976

[2] E. Bertino, S. Jajodia, L. Mancini, and I. Ray, Advanced Transaction Processing in Multilevel Secure File Stores, IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 1, 1998, pp. 120-135.

[3] H. Schwetman, CSIM User's Guide for Use with CSIM Revision 16, Microeletronics and Computer Technology Corporation, 1992.

[4] J. McDermott and S. Jajodia, "Orange Locking: Channel-Free Database Concurrency Control Via Locking," in B.M. Thuraisingham and C.E. Landwehr, ed., Proc. IFIP WG11.3 Working Group on Database Security, North-Holland, 1992, pp. 267-284.

[5] K.P. Smith, B.T. Blaustein, and S. Jajodia, Correctness Criteria for Multilevel Secure Transactions, IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 1, 1996, pp. 32-45.

[6] L.V. Mancini and I. Ray, Secure Concurrency Control in MLS Databases with Two Versions of Data, Proc. European Symp. Research in Computer Security,.Rome, Italy, Sept. 1996, pp. 204-225.

[7] P. A. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987, pp. 25 – 45, pp. 143 – 166.

[8] R. Agrawal, M. J. Carey, and M. Livny, "Concurrency Control Performance Modeling: Alternatives and Implications," in Kurmar, V. ed., Performance of Concurrency Control Mechanisms in Centralized Database Systems, Prentice Hall, 1996, pp. 58 – 105.

[9] S. Jajodia, L. V. Mancini, and I. Ray, Secure Locking Protocols for Multilevel Database Management Systems, Proc. IFIP WG11.3 Working Group on Database Security, Como, Italy, July 1996, pp. 177-194.

[10] T. F. Keefe and W.T. Tsai, Multiversion Concurrency Control for Multilevel Secure Database Systems, Proc. IEEE Computer Society Symp., Research in Security and Privacy, Oakland, California, U.S.A, May 1990., pp. 369-383.

[11] V. Atluri, S. Jajodia, and T.F. Keefe, Multilevel Secure Transaction Processing: State and Prospects, Proc. IFIP WG11.3 Working Group on Database Security, Como, Italy, July 1996, pp. 79-98.

[12] W. T. Maimone and I.B. Greenberg, Single-Level Multiversion Schedulers for Multilevel Secure Database Systems, Proc. 6th Annual Computer Security Applications Conference, Tucson, Arizona, U.S.A., Dec. 1990, pp. 157-180.

[13] Y. Sohn, Confidential Concurrency Control for Secure Transaction Management in Database Systems: C3, Ph.D. Thesis, KAIST, Seoul, Korea, 2000, pp. 120 – 127.

Task-Role Based Access Control (T-RBAC): An Improved Access Control Model for Enterprise Environment

Sejong Oh [1], Seog Park [1]

[1] Sogang University, Dept. of Computer Science,
121-742, Seoul, Korea
{sejong, spark}@dblab.sogang.ac.kr

Abstract. Companies today manage business information with computer systems and many users access it for their job functions. Companies need security mechanisms to effectively protect important information. Moreover they need to minimize the interruptions from security mechanisms that cause delays in the execution of business activities. It is a difficult problem. In this paper we analyze the requirements of access control in enterprise environment and propose classifications for job functions. We propose a improved access control model for enterprise environment through integration of role based access control and activity based access control model.

1 Introduction

Since many companies recognized that the computer is an important means for increasing their competitive power, they have competitively built computer systems. Growth of companies, volumes of information, and related personnel have increased. As a result, security problems have become increasingly difficult.

The method of information protection is authentication, authorization, access control, audit, encryption, etc. In this paper we focused on access control. Access means the ability to perform work such as reading, writing, and the execution of the system resources. Access control is the way to control the ability for performing the work. Access control of the computer system describes whether specific users or processors can access specific system resources or not, and their allowed access type.

Researchers have developed some access control models such as discretionary access control, mandatory access control, and role based access control. An activity based access control model was introduced recently. It was designed for collaborative work environment. Because these models have constraints on applications for enterprise environment, it is necessary to investigate a proper model for enterprise environment.

The purpose of this paper is to propose a proper access control model for enterprise environment through the integration of the role based access control and activity based access control models. The rest of this paper is organized as follows. Section 2 re

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 264-273, 2000.

views the characteristics of enterprise environment related to access control and shows access control requirements on enterprise environment. Section 3 reviews related investigations and shows their limitations for enterprise environment. Section 4 introduces a new access control model, the task-role based access control. Section 5 discusses advantage points of our new model. Section 6 presents the conclusion and proposes further work.

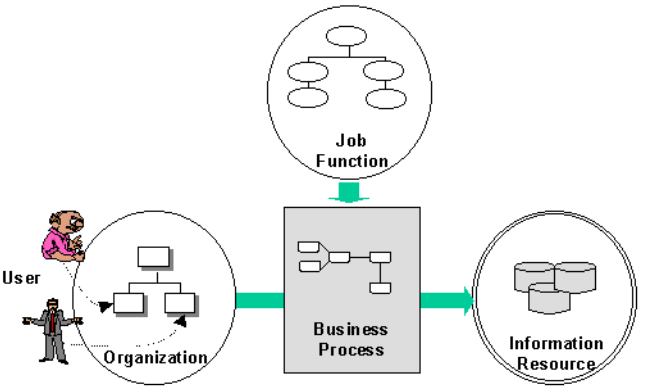


Fig. 1. Enterprise Environment

Table 1. Classification of job functions.

Classification of Job functions		Class id	Example job functions	Characteristics
Supervision job functions		Class S	✔ supervise ✔ review ✔ delegation	✔ has a access right inheritance ✔ access right hierarchy is similar to organization hierarchy
Essential job functions	Workflow oriented job functions	Class W	✔ drafting & approval ✔ chained job *(see Fig.3)*	✔ belongs to workflow ✔ needs active access control ✔ needs many integrity rules (ex. Separation of duty)
	Non workflow oriented job functions	Class P	✔ analysis ✔ planning ✔ decision making	✔ private job function (has no relationship with other job functions or other job positions)

2 Requirements of Access Control on Enterprise Environment

2.1 Classification of Job Functions

The basic goal of access control is to offer a methodology that only authorized users (subject) can access information resources (object). From an access control point of

view, enterprise environment is expressed in Fig. 1. In general, users in the company belong to the organization structure and are performing their assigned job functions according to their job positions. Job functions in company compose business process. Users read or write information objects for executing their job functions. Some of these information objects are important ones that should be protected from general users, and access should be allowed only by the users who are directly involved with the work. Therefore job functions of a user are the bases of access control. By observation of the enterprise environment, we found that there are three classes of job functions such as in table 1. If a user U_1 has job functions that belong to class S, their related access rights are inherited to user Un who has a higher position than U_1 in the organization structure. But class W and class P do not have such inheritance characteristics. Job functions belong to class W, which has a relation with workflow and show the characteristics of an active security model. Access control of the enterprise environment needs a proper method to deal with three classes of job functions through different ways. Our suggested model is based on the classification of job functions. In section 2.2, we detail requirements of access control in the enterprise environment. (Sometimes job function is called 'task'. So we use 'job function' and 'task' interchangeably).

2.2 Requirement of Access Control

The characteristics of access control in the enterprise environment lead us to security requirements as follows.

(Req.1) A user should have an access right only to information objects that are need for doing his/her job functions. (Principle of least privilege).

(Req.2) General users cannot discretionary change security attributes, for example access rights, of information objects. Changing security attributes may induce an outflow of information. Only security administrator can do it.

(Req.3) The organization structure of a company has a deep relationship with authorization and supervision structure. In general, the higher position in an organization structure, the more authority and responsibility. Some authority of lower position is inherits to higher position automatically. So access control model reflects organization structure and its characteristics.

(Req.4) There are some of job functions for a job position that are essential and don't be inherited to its higher job position. For example, *manager* is a higher job position than that of *clerk*, however, manager doesn't automatically inherit the *'register purchase'* job function of *clerk*.

(Req.5) Some job functions belong to workflow in the company and their related access rights should be available during the allowed execution of those job functions. (See Fig. 3).

(Req.6) The access control model for enterprise environment should be able to deal with many users and many information objects. So the design for authorization, assignment of access rights, and change of them must be easy. (For example, access matrix or access control list[2] are not proper to enterprise environment).

(Req.7) The execution of access control should not make for inconvenience and a lowering of efficiency for the execution of job functions.
(Req.8) The access control model should support integrity principle such as separation of duty or delegation of authority.

3 Related Work

In this section, we review related investigations with access control and analyze their limitations them when they are applied to the enterprise environment and also we review discretionary access control (DAC), mandatory access control (MAC), role based access control (RBAC), and activity based access control (ABAC).

Discretionary access control (DAC).[1], as its name implies, access control depend on discretion of object's owner or anyone else who is authorized to control the information object's access. The obvious advantage of DAC is that users are afforded great flexibility. However, it is difficult for DAC to guarantee integrity rules such as 'least privilege' and 'separation of duty', that are needed for the enterprise environment. DAC is proper to the environment when information sharing is more important than protection of information. (DAC doesn't guarantee requirement (Req.1), (Req.2), and (Req.5) in section 2. DAC doesn't cover the issues (Req.3) and (Req.4)).

Mandatory access control (MAC).[2] means that access control policy decisions are made beyond the control of the individual owner of object. A central authority determines what information is to be accessible by whom, and the user cannot change access rights. In the MAC model, users (subjects) and information objects have labels by their security level, and users are restricted access according to their security level. MAC protects against information disclosure. But in the enterprise environment, security labeling is difficult and it is not convenient for job execution. So MAC is not proper for the enterprise environment. (MAC doesn't guarantee requirement (Req.3), (Req.4), (Req.5), and (Req.7)).

Role based access control (RBAC).[6][7] has the central notion of preventing users from accessing company information discretionarily. Instead, access rights are associated with roles, and users are assigned to appropriate roles. The notion of role is an enterprise or organizational concept. As such, RBAC allows us to model security from an enterprise perspective since we can align security modeling to the roles and responsibilities in the company. This greatly simplifies management of access rights. Fig. 2 shows the basic components of RBAC and a sample of a role hierarchy. In the real world, a role can be defined as a job function within the organization that describes the authority and responsibility conferred on a user assigned to the role. Role hierarchies are natural means for structuring roles to reflect an organization's lines of authority and responsibility. It is defined as partial order relationship of related roles. As role hierarchies are similar to authorization system, it is suitable for modeling of enterprise organization structure. RBAC model has requirement limitations as outlined in section 2 as follows.

– Table 1 shows that company has three types of job functions and they need different access controls. But RBAC doesn't consider this point.
– Basic RBAC model has a role hierarchy concept that higher role inherits all access rights of lower role in the role hierarchy. But access right assigned essential job function, in the table 1, should not be inherited to its higher role in the real world. (Req.4).
– RBAC doesn't consider workflow. Workflow needs a dynamic activation of access right and specification of application level constraints [3]. (Req.5).

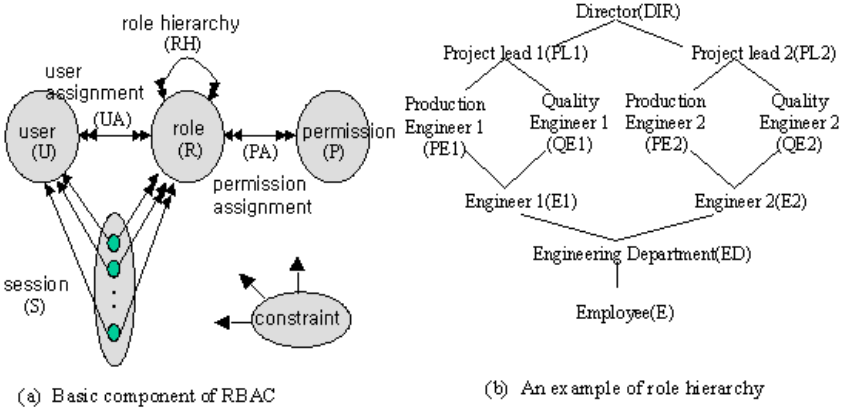


Fig. 2. Basic RBAC model

Activity based access control (ABAC).[4][10][11] is investigated for a collaborative work environment represented by 'workflow'. Workflow is defined as a set of activities that are connected to achieve a common goal. There are many workflows in the company. Each workflow produces its instances. Fig. 3 shows an example of workflow. ABAC separates access right assignment for users and access right activation. Even if a user was allocated access right related to his/her task, he/she can exercise his right during the activation of the task. Also ABAC offers specifications for the application level constraints. Moreover, it supports the implementation of integrity rules of the real world. Many researchers agree with the fact that concept of role is useful for workflow environment. Some researchers investigated development of specification language for expressing application level constraints that related to the role and workflow [3][8][9]. ABAC has limitations in the enterprise environment as follows

– There exist many tasks that don't belong to workflow in the company, and ABAC doesn't deal with them. So extra access control methods should be added to ABAC. (Req.4~6).
– In the real world, a superior officer supervises and reviews execution of tasks of his/her inferior clerks. It's important for security and integrity; however, ABAC doesn't take review and supervision into consideration. (Req.3).

In this section, we review related work with access control models. Even though each model has its good points, there are limitations of application in the enterprise

environment. So we propose new access control model, task-role based access control (T-RBAC), which is more efficient for enterprise environment. Our strategy is to integrate RBAC and ABAC, and apply task classification to T-RBAC. Already there are many investigations for the specification of constraints; however, we don't deal with it. In the next section, we describe T-RBAC.

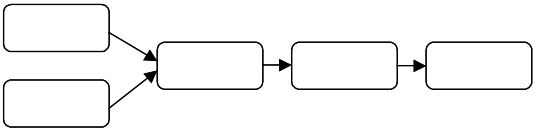


Fig. 3. Example of workflow

4 Framework for Task-Role Based Access Control Model

The T-RBAC, enhanced model for enterprise environment, is an integration model of the role based access control (RBAC) model and activity based access control (ABAC) model. Fig. 4 shows a brief of T-RBAC. The most difference between T-RBAC and RBAC in Fig. 2 is that the access rights are assigned to task in T-RBAC, rather than access rights are assigned to role in RBAC. In the real world access rights are needed for the user to perform tasks. So assignment of access rights to task is reasonable. We analyze the effect of the assignment of the access rights for tasks in Section 5. In T-RBAC, tasks are classified into three classes, and are treated differently. Tasks in Class W are used to compose workflow. Workflow creates the instances that are set of task instance. In the real world, a company has many workflows, but we assume a single workflow for simplicity. Now we will define some notations and explain characteristics of T-RBAC by properties. The concept of session and user-role assignment (URA) follows RBAC.

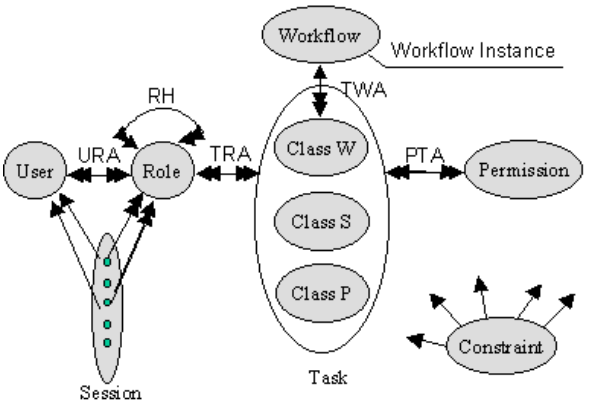


Fig. 4. Task-Role Based Access Control (T-RBAC) model

We use some notations for properties, as follows.

U : a set of users	RH : role hierarchy
R : a set of roles	P : a set of permissions
S : a set of sessions	Ri < Rj : Rj is a higher role of Ri on RH
W : workflow	Ti →Tj : Ti is a prior task of Tj on W

Property 1. Classification of tasks. All of tasks are classified into 3 classes as follows:

- Class S : supervision oriented tasks
- Class W : workflow oriented tasks
- Class P : private tasks

Tw , a set of tasks in Class W, Ts, a set tasks in Class S, Tp , a set of tasks in Class P :
A set of task $T = Tw \cup Ts \cup Tp$ and $Tw \cap Ts = \varnothing$, $Ts \cap Tp = \varnothing$, $Tw \cap Tp = \varnothing$

Property 2. Task-Role Assignment. A Role has tasks one or more. A task can be assigned to many roles.

$TRA \subseteq R \times T$

Property 3. Permission-Task Assignment. A task has permissions for executing itself. Permission is defined as a pair of information objects and access mode.

$PTA \subseteq P \times T$

Property 4. Role hierarchy and partial inheritance. IF R_1 is a higher role than R_2 in a role hierarchy, R_1 inherits only the tasks of R_2 that are belong to Class S.

$R_1, R_2 \in RH : R_1 > R_2 \Rightarrow R_1$ inherits $\cup$ (Ti), $Ti \in Ts \wedge Ti \in R_2$

Property 5. Task-Workflow Assignment. Only the tasks that belong to Class W can be assigned to workflow.

$Ti \in W \Rightarrow Ti \in Tw$

Property 6. Task attributes. If task Ti belongs to workflow, Ti has three attributes as follows:

- activation condition (AC) : it's an activation condition of Ti. It is expressed AND/OR condition between prior tasks. For example, if AC = T2∧T3, Ti can be activated when prior tasks T2 and T3 are finished.
- activation flag (AF) : it tells Ti is activated or not. If Ti is deactivated, AF is null. If Ti is activated, AF contains the time that Ti is activated.
- available duration time (AT) : it's an effective execution time of Ti. If AT is over after AF, Ti is deactivated automatically.

These attributes are inherited to new task instances when they are created in new workflow instances.

Property 7. Execution order. IF $T_1,T_2,..,T_n$ are prior tasks of Ti and Ti is activating now, $T_1,T_2,..,T_n$ are deactivated automatically.

{ $T_1,T_2,..,$Tn} fi Ti : activate(Ti) deactivate($T_1,T_2,..,$Tn)

Property 8. Access right activation. If Ti belongs to workflow, the access rights assigned to Ti can be allowed to user only for Ti is activated.

Pi ¸ P, Pi ¸ assigned(Ti) : Ti ¸ Tw activate(Pi) only if activate(Ti)

Property 9. Separation of duty. T-RBAC offers task and instance level separation of duty policy. Instance level separation of duty policy apply to tasks belong to Class W, and it is effective for tasks belongs to same workflow instance. For example, if task Ti and Tj are mutually exclusive at instance level and instance of Ti and Tj belong to different workflow instance, separation of duty policy doesn't apply to Ti and Tj.

- [TS-SOD] task schema level static separation of duty
  Ui ¸ U, Ri ¸ R, Ti, Tj ¸ T, Ti „ Tj :
  Ti ¸ assigned(Ri) Tj ¸ assigned(Ri) Ri ¸ assigned(Ui)
  Ti ˇ mutually-exclusive-task(Tj)
- [TD-SOD] task schema level dynamic separation of duty
  Ui ¸ U, Ri ¸ R, Ti, Tj ¸ T, Ti „ Tj :
  Ti ¸ assigned(Ri) Tj ¸ assigned(Ri) activate(Ti) activate(Tj)
  Ri ¸ assigned(Ui) Ti ˇ mutually-exclusive-task(Tj)
- [ID-SOD] task instance level dynamic separation of duty
  Ui ¸ U, Ri ¸ R, Ti, Tj ¸ Tw, Ti, Tj ¸ same instance of W, Ti „ Tj :
  Ti ¸ assigned(Ri) Tj ¸ assigned(Ri) activate(Ti) activate(Tj)
  Ri ¸ assigned(Ui) Ti ˇ mutually-exclusive-task(Tj)

[Note] In the property 6,7, and 8, task means task instance in the workflow. Instance level separation of duty has a same meaning and effect with task level static separation of duty.

5 Discussion

There are two central ideas in the T-RBAC. One is the classification of enterprise tasks (job functions) according to their characteristics. The other is to use intermediate tasks between access rights and roles instead of assigning access rights to roles. It makes possible that roles can be linked to access rights through intermediate tasks. Moreover, it makes the point of contact that RBAC could be integrated to ABAC model. Fig. 5 shows access control feature in T-RBAC model. A user can run various tasks through assigned roles. Some tasks (belong to class P) are inheritable and their access rights are inherited to higher roles in the role hierarchy. Some tasks (belong to class S) are private and don't be inherited to higher roles. Some tasks (belong to class W) follow active security policy and they are managed by workflow mechanism.

The classification of tasks (job functions) has following effects by their characteristics.

- It is simplified that which task/access rights can be inherited to higher roles from

lower roles. (Only tasks belong to class S has an inheritance characteristic).
- It is possible that we apply the ABAC model (*active security model*) to tasks belong to class W and apply the general *subject-object security model* to tasks belong to class S or P.

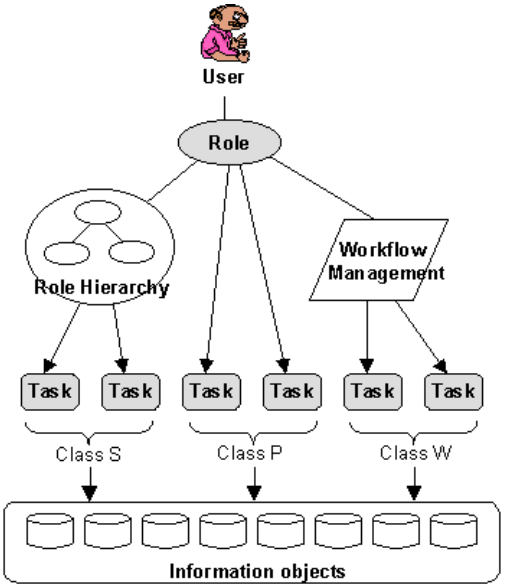


Fig. 5. Access Control on T-RBAC model

Also, the method in which access rights are linked to roles through tasks has following effects as compared with general RBAC model.
- In the real world, access rights are given to roles according to roles' tasks. However, in the general RBAC model, it cannot be known that what access rights are needed to perform given role's task. So it is difficult to modify access rights in case that role's task is changed. The new method not only solve above problems but also makes it possible that delegations can be made on the bases of task-based unit instead of role-based unit.
- In the RBAC model, the unit of separation of duty is a role. But in the T-RBAC, the unit is task. Task unit has more small scope of access rights than role unit. So T-RBAC can support more elaborate separation of duty policy.

The integration of RBAC model and ABAC model has following effects.
- It can deal with the workflow oriented tasks.
- It can deal with non-workflow oriented tasks (Class S, Class P).
- It can solve supervision problem of ABAC model by defining supervision tasks (Class S) for workflow oriented tasks (Class W).

Considering above discussion, T-RBAC model solves the limitations of ABAC model and RBAC model presented in chapter 3. Therefore, we can say that T-RBAC model is more suitable to enterprise environment than general RBAC model or ABAC model.

6 Conclusion and Further Work

To develop proper access control mechanism for enterprise environment is very difficult because of numerous users, numerous information objects, various kinds of security policies such as subject-object security and active security, organization structure, and inter-relationship of business processes. So general access control mechanisms – DAC, MAC, RBAC, and ABAC – have limitations to apply to enterprise environment.

In this paper we propose more efficient access control model, T-RBAC, for enterprise environment by classification of job functions (tasks) and integration of RBAC and ABAC. Main contribution point of this paper is as follows. First, we analyze access control environment of enterprise and define security requirement of access control area of enterprise environment. Second, we classify job functions into three classes and apply different access control mechanism to them. Third, we propose T-RBAC model, which is integration of RBAC and ABAC. T-RBAC satisfies eight security requirements in section 2.

T-RBAC is useful for large companies that have complex organization and business process. We have an investigation plan for delegation on T-RBAC and extend T-RBAC model to distributed environment.

References

1. C.P.Pfleeger: Security in Computing, second edition, Prentice-Hall International Inc.(1997)
2. E.G.Amoroso: Fundamentals of Computer Security Technology, PTR Prentice Hall (1994) 253-257
3. E. Bertino, E.Ferrari, V.Atluri: A Flexible Model Supporting the Specification and Enforcement of Role-based Authorization in Workflow Management Systems, Proc. of 2nd ACM Workshop on Role-Based Access Control (1997)
4. Dagstull, G.coulouris, J.Dollimore: A Security Model for Cooperative work : a model and its system implications, Position paper for ACM European SIGOPS Workshop, September (1994)
5. R.S.Sandhu, P.Samarati: Access Control : Principles and Practice, IEEE Communication Magazine, Sep. (1994) 40-48
6. R.S.Sandhu, E.J.Coyne, H.L.Feinstein, C.E.Youman: Role-Based Access Control Method, IEEE Computer, vol.29, Feb. (1996)
7. D.Ferraio, J.Cugini, R.Kuhn: Role-based Access Control(RBAC): Features and motivations, Proc. of 11th Annual Computer Security Application Conference, Dec.(1995)
8. W.K.Huang, V.Atluri: SecureFlow: A Secure Web-enabled Workflow Management System, Proc. of 4'th ACM Workshop on Role-Based Access Control (1999)
9. M.S.Oliver, R.P.Reit, E.Gudes: Specifying Application-level Security in Workflow Systems, Proc. of 9'th International Workshop on Database and Expert Systems Applications (1998)
10. R.K.Thomas: Team-based Access Control(TMAC):A Primitive for Applying Role-based Access Controls in Collaborative Environment, Proc. of 2nd ACM Workshop on Role-Based Access Control (1997)
11. R.K.Thomas, R.S.Sandhu: Task-based Authorization Controls(TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management, Proc. of the IFIP WG11.3 Workshop on Database Security (1997)

Data Security for Distributed Meeting Systems

Ryong Lee, Yahiko Kambayashi

Department of Social Informatics, Kyoto University
Yoshida-Honmachi, Sakyo, Kyoto, 606-8501 Japan
{ryong, yahiko}@isse.kuis.kyoto-u.ac.jp

Abstract. In a real world meeting, sometimes it is required to use secret documents which are used during the meeting, but are not permitted to bring back. Such a function is very hard to be realized by distributed meeting systems consisting of computers and networks. Participants of the meeting cannot usually access the material after the meeting. The requirements of such security meeting are contrary to the open connectivity of distributed environments. Thus, the security meeting remains in old styles that the participants come together at a restricted meeting room with their authorizations. In this paper, we discuss constraints to use the secret material on a distributed security meeting, and propose the strictly restricted meeting system which controls action and state of meeting participants with supporting multi-level security. The architecture of the proposed system is called **SDMS(Secure Distributed Meeting System)**, and its requirements to be realized on the Server-Client model are described.

1 Introduction

Due to the recent development of distributed computing environment, the style of performing work is changing rapidly. As a result, a lot of work could be free from restrictions of time and space. However, some work remains in traditional styles, which are not suitable for distributed environment because of practical reasons. One typical case is a meeting with secret data which can be used only during the meeting. In addition, the participants cannot access the material after the meeting. Requirements of such a security meeting are summarized as the followings:

- **Meeting Place**
  restriction of time and space to the meeting participants
- **Meeting Entrance**
  restriction of participating in the meeting by unauthorized peoples
- **Secret Material**
  security data management from its creating to deletion

These requirements for opening the secure security meeting are inconsistent with general characteristics of computing environment distributed over distinct

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 274–286, 2000.

places and different working time. The first requirement, the restriction of meeting places has been applied historically to prevent illegal actions of the participants. The distributed meeting, however, already has been thought to be useful due to the various advantages. Thus, we need to consider a possibility to take place such security meeting on the distributed environment.

The second requirement is how to check rights authorized to participants. In current computer technologies, there are basic solutions using IDs and passwords with cryptography, IC-Cards, or recognition of body features, etc. By using these methods, authorization on the distributed environment can be confirmed safely in general. The last requirement for managing the secret material requires the most complicate processing. In the traditional meeting(without computers), secret data are distributed to the specific participants in a restricted room, and after meeting, the data must be abolished immediately. This method ensures the data are given to the authorized participants, and returned to the distributor safely.

If we take place the meeting on the distributed computer environment, the following conditions must be satisfied:

Restricted Communication :
The communication line between the data distributor and participants must be secure at least during the meeting.

Restricted Access :
Only during the meeting, the distributed data at each participant must be accessed with his/her authorization.

Many security studies have been focused on the secure communication condition. However, the second condition which requires to control resources of participant machines becomes to be an important consideration[6][8].

In this paper, we will discuss constraints of secure data distribution for the distributed security meeting, and a system supporting such a function.

2 Constraints to open the distributed security meeting

In order to guarantee the secure meeting on the distributed environment, we assume the following two constraints related to action and state of participants. The action is an activity of a participant in accessing the secret material, and the state is a situation that the participant has the material. With the two controls, the meeting manager is able to open the security distributed meeting since the participants are reliable.

The following two constraints correspondent to the two controls, the action and the state of a participant.

2.1 Meeting Place

This constraint has two means: space and time restrictions. We here consider two types of space restrictions.

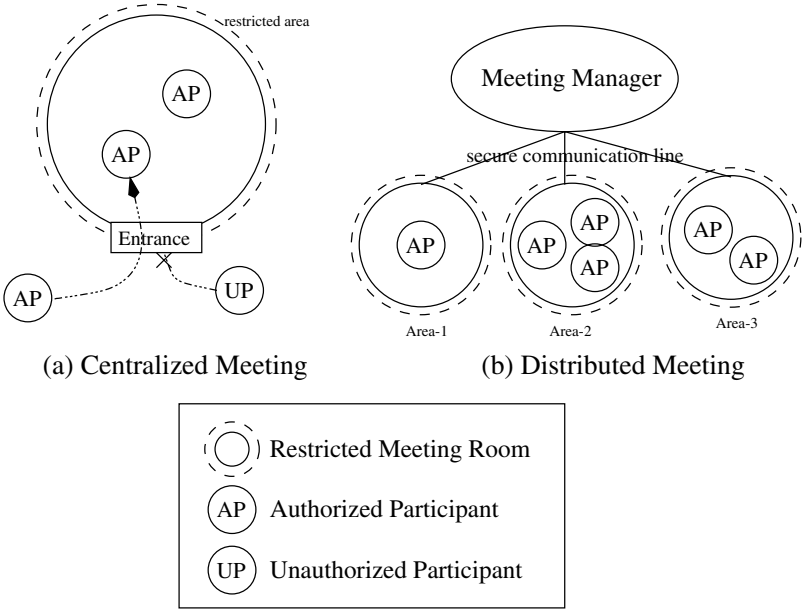


Fig. 1. Meeting Type : Centralized, and Distributed

- **strictly restricted**
  We can think a case, for example, a company has several branches, and takes place a security meeting between their branches in the restricted room each other. Each branch must be ensured that it has a restricted place and a restricted machine generally. Meeting manager of the meeting controls all resources of them. Participants can use each machine just only during the meeting time. In addition, each meeting area needs to be monitored by video camera. For accurate and efficient surveillance, the monitoring method using the special camera like omnidirectional camera[7] may be a good solution.
- **loosely restricted**
  Participants sometimes want to join the meeting with his/her own machine as like a personal desktop or a mobile computer. The method that sends a secret material to the participant machine safely can be implemented with cryptography generally. However, the control of resources on the machine is almost impossible currently because of sufficient or unreliable control mechanisms to private machines. The least resource controls to the target machine are requested in return for downloading the secret material. That is, existence of the data and its accessibility on the participant machine must be made clear in the data distributor, with secure usages of the machine physically. Of course, it is very hard to prevent from using a camera to copy the display contents without notice by the meeting manager. We can assume that there is a video camera to watch each participant. In the following, we

assume that it is not possible to copy the display contents by an equipment not controlled by the system.

2.2 Secret Material

Secret data usable on the distributed environment
Before further discussions, we need to set a boundary at the secret data to be used for distributed meeting. Generally, secret materials are not always usable on the distributed environment. Thus, it is reasonable that top-secret materials are better to remain the old centralized style practically, which is accessible in a specialized room just for reading.

We now consider the security level of secret data in a company as the followings:

- **Inside-Company** : Data items which must be limited for its using just only inside of the company from its content. Nevertheless, a low privileged group can access it for the working in the company.
- **Inside-Department** : Data items which must be limited for its using just only inside of department from its content. In many cases, it needs to be encrypted. A group who needs it for their affairs can access it.
- **Top-Secret** : Data items which must be kept in the high security and to be encrypted essentially. The restricted group only can access it with logging the uses.

In this paper, we will consider the first two cases, secret data of Inside-Company, and Inside-Department. It is possible to generalize the case by introducing a complicated hierarchy of the organization, but we will not discuss such a case, since the basic idea is same. By using such secret data on the distributed environment, we can get more advantages that are economical relatively by permitting it. However, the last case, Top-Secret data have no realistic advantages yet since there still exists very high risks if it would be disclosed.

Management of secret material
Keeping watch the distributed secret material is important problem with keeping secure communication. In using a secret material on the distributed security meeting, the following constraints must be considered.

Key Possession
For safe communication, we need to encrypt the secret material. Here we consider Public-Key Infrastructure[3]. Encrypted secret material is decrypted on the participant machine after confirming the authorized user, and then the authorized user can get the access. In PKI mechanism, however the key to decrypt must be shared to the participant with secret data. In our security meeting condition, participants must be forbidden in accessing the secret material after meeting. Thus, general PKI mechanism is not sufficient to prevent participants from having the secret data after meeting. If possible,

the secret data on the participant machine should be decrypted not showing the private key directly. Such trial was done practically in [4], where instead of sending private key directly to users, a token function is used to decrypt the secret material with private key indirectly.

Security Level

Secret material may have its own security level from low to high. In our discussion, In-Company secret data have low security level, and In-Department secret data have high security level. Secret data are only able to access by authorized users who are higher privileged than security level of the data. If the secret material is modified during the meeting, a new security level of it should be higher than the old level[5].

Lifetime

Only during the meeting, participants can access secret material by a right authorization. To conform the time on the meeting participant machine, we needs to synchronize the time of participant machine periodically or to use relative time, for example, 'during 10 minutes'. After the meeting, if possible, the secret material is desirable to delete completely on the participant machine.

For above constraints to ensure secure security meeting, the meeting manager needs to control the participants machine on the distributed environments. In the following two sections, we suggest a strictly restricted type model on the Server-Client model to realize the distributed security meeting system.

3 Secure Distributed Meeting System(SDMS)

A powerful way to forbid users to commit illegal actions at the secret data or to enable restricting actions may be that the machine has little number of I/O interfaces as possible as physically(for example, network computers). There is some approach that the machine has only a monitor, a keyboard or a touch-screen, a mouse, etc. to suppress a user action in general [1]. These approaches might be successful for a strong security system, but not so efficient, realistic, and extensible on the distributed environments. Really, we often take place heterogeneous meetings in a viewpoint of security level. Therefore, a more flexible security system is needed with supporting strong security only in an important secret meeting.

For realizing such heterogeneous security meeting, we will propose a security meeting system supporting multilevel security in the client-server model. To implement strictly restricted security meeting, we must control all the resources of the machines (clients) which users use on the distributed environment. The resources include I/O interfaces, applications, network, and working data.

The security level of meeting can be divided into tree levels following:

Low-Security Level Meeting

A user can utilize the resources freely, that is a user can import or export data by I/O interfaces, execute any applications for manipulating the data,

and access network from local network to external network for another data that are more plentiful supporting. As results, users can make profits as the freedom of meeting activity.

Middle-Security Level Meeting

In this level, the meeting manager at the server determines the restrictions of each resource of clients specifically for heterogeneous meeting types.

High-Security Level Meeting

Strong restrictions are applied to the resources, especially at the working data. These data can be accessed in the secret meeting, and then is desirable not to export to the external of the meeting. In our system, the working data should be deleted after meeting.

We can now simulate all heterogeneous security meeting in this model. In the following sections, we describe our system architecture supporting above multilevel security meeting.

3.1 System Architecture

Secure Distributed Meeting System(SDMS) is constructed on the client-server model. Server is for the meeting manager, and client is for users, that is, server controls all actions and states of clients.

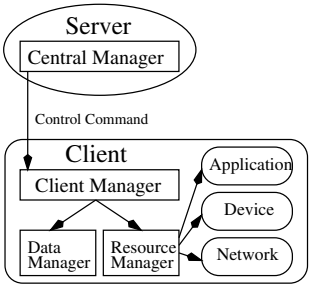


Fig. 2. SDMS Architecture

In Fig.2, Meeting manager can control clients by the 'Central Manager'. Client Manager controls all resources of clients by a mandate of server. This manager specifically consists of two functions, 'Resource Manager' and 'Data Manager'.

Resource Manager

This function controls permission for users to access devices, applications, and network. If each resource receives no permission from server, a user cannot utilize any resources.

Device Control

The 'Device' generally includes all interfaces for data input or output. In our model, we divide devices as two sets, 'Read' devices and 'Write' devices. Read device is for importing data from external to internal of computer, as like keyboard, mouse, FDD, CD-ROM, and etc. Write device is for exporting data. Read device can be utilized on a security meeting, but write device cannot. The devices like FDD that are included in two sets simultaneously can be divided by its permission as the concept of Unix file permission. In the low-level security, we can export data at FDD, but just only import data from FDD in high-level security meeting.

Application Control

Application for manipulating data is also controlled by Resource Controller. The restriction of application can be realized in two methods [5].

- Control Execution of Applications
- Control Ability of Applications

The first method makes the security system to be application-independent, so that it makes system management easy and users get more manipulating functions with rapidly increasing data format. In low-security level, we can execute all applications without restrictions. However, some applications are not available freely for preventing users from correcting or exporting the secret data in high-security level, or for concentrating users on the specific applications.

Network Control

Lastly, network device is the most risky part in security meeting systems. The complete restriction of network access is not good to realize efficient meeting. Therefore, we should better forbid the usage of network only when the high-security is required. In low-security level users can connect to local area network or to external network directly. Moreover, in middle-security meeting, a user can connect the external network only through the server indirectly.

In these three controls, meeting manager can control the action by a user on the client. Participants cannot export data to external network without the permission of server in a security meeting.

Data Manager

After the meeting, the secret data on the client are remained. For more secure security management, the data must be deleted completely. If changes occur in the data, it needs to be returned to server safely. The data generally are not only the secret data but also the user-written data, or cached data. In many cases, the secret or cached data are also desirable to be deleted completely.

Our system architecture as above is ensured for the meeting manager to control the actions and states on the clients perfectly. In addition, it simulates the

first strong security model by Resource Manager and Data Manager in high-security level. With the two basic security levels, high- and low-, the middle security level is also applicable in many heterogeneous meetings. Chairman of meeting can constitute different security meeting with options on Resource Manager and can distribute a secret data safely on Data Manager. A user also can utilize more resources on the meeting for better meeting results.

3.2 Role-Based User Access

In the above multi-security meeting system, meeting manager controls state and actions on all clients. However, real meeting are held with participants who have complex relationships of role. We therefore must consider such semantic elements about role of meeting among chairman, observer, normal participants, etc. An action of a participant generally can be restricted from his or her role on the meeting. Thus, a participant who has a high privilege becomes to be less restricted.

In general RBAC (Role-Based Access Control) model[2], permission to the resources is allocated to the role, and users are members of the role. This concept simplifies security system implementation and administration.

In our model, we would support a role based access on above the multi-level security meeting system controlled by the meeting manager from its action and state.

4 Meeting Description

Now we define **Meeting Session** for further discussions.

Definition 1. Meeting Session (MS) is an object to manage a security meeting with the following properties:

- security : Security level of meeting (high, middle, low)
- duration : Duration time of meeting
- role : Role Manager

Definition 2. Role Manager (RM) is an object to manage user-role-client(terminal) relation with the following properties:

- id : List of IDs
- privilege : Privilege level of role
- duration : Lifetime of the meeting
- terminal : Terminal Object

Definition 3. Terminal Manager(TM) is an object to interactive with clients directly with the following properties:

- device :
  - controltype : Control type of each device : Read, Read&Write, Write
  - controlstate: Control state
  - duration : Access time
- application :
  - controltype : Control by execution, or ability
  - controlstate : Control state
  - duration : Access time
- network :
  - controlarea : Reachable Area (LAN, or External Network)
  - duration : Access time
- duration : Lifetime of the terminal

We can now describe multi-security meeting with these definitions. Firstly, a meeting that has high-security level is described as like:

```
MO_1.security=High
MO_1.duration=2000/1/24/11:00-12:00
MO_1.role[chairman]
MO_1.role[participant]
```

Here it describes the meeting session MO_1 has the high security level, and lifetime(duration), and role definitions. Then users are allocated to the roles and security, duration, and terminal of each role are ready.

```
MO_1.role[chairman].id[]=Kambayashi
MO_1.role[chairman].security=high
MO_1.role[chairman].duration=MO_1.duration
MO_1.role[chairman].terminal[]
MO_1.role[participant].id[]=user1,user2,user3
MO_1.role[participant].security=middle
MO_1.role[participant].duration=MO_1.duration
MO_1.role[participant].terminal[]
MO_1.role[observer].id[]=user4
MO_1.role[observer].security=low
MO_1.role[observer].duration=2000/1/24/11:30-11:40
MO_1.role[observer].terminal[]
```

In next description, we can see the terminal specification for each role. The chairman can use the terminal id[0], and its duration is equal to his role duration, and also the participant.

```
MO_1.role[chairman].terminal[0].device
MO_1.role[chairman].terminal[0].application
MO_1.role[chairman].terminal[0].network
MO_1.role[chairman].terminal[0].duration=
     MO_1.role[chairman].duration
MO_1.role[participant].terminal[1].device
MO_1.role[participant].terminal[1].application
MO_1.role[participant].terminal[1].network
MO_1.role[participant].terminal[1].duration=
     MO_1.role[participant].duration
```

Now chairman named *Kambayashi* has all controls his terminal, and thus is able to take all free actions. And the participant named *user1* who is using terminal[1] can read data from floppy device.

```
MO_1.role[chairman].terminal[0].device[*].controltype=*
MO_1.role[chairman].terminal[0].device[*].controlstate=*
MO_1.role[chairman].terminal[0].device[*].duration=
     MO_1.role[chairman].terminal[0].duration
MO_1.role[participant].terminal[1].device[floppy].controltype=RW
MO_1.role[participant].terminal[1].device[floppy].controlstate=ReadOnly
MO_1.role[participant].terminal[1].device[floppy].duration=
     MO_1.role[participant].terminal[1].duration
```

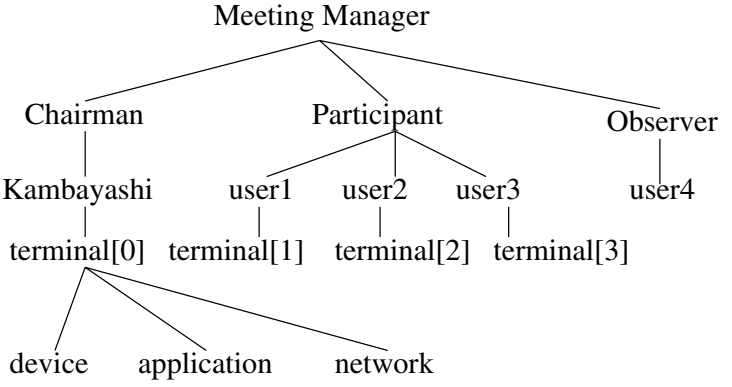


Fig. 3. Meeting Configuration

The following describes that the browser application can be executed by the participant who uses the terminal[1].

```
MO_1.role[participant].terminal[1].application[browser].controltype=
     execution
MO_1.role[participant].terminal[1].application[browser].controlstate=
     executable
MO_1.role[participant].terminal[1].application[browser].duration=
     MO_1.role[participant].terminal[1].duration
```

Lastly, the available network area is described. The chairman can use any network access, LAN or External Network, but the participant just can use LAN access.

```
MO_1.role[chairman].terminal[0].network.controlarea=*
MO_1.role[chairman].terminal[0].network.duration=
     MO_1.role[chairman].duration
```

```
MO_1.role[participant].terminal[1].network.controlarea=LAN
MO_1.role[participant].terminal[1].network.duration=
     MO_1.role[participant].terminal[1].duration
```

4.1 Meeting Session

The meeting session is created with pre-determined configuration as the above descriptions, and goes through the next steps.

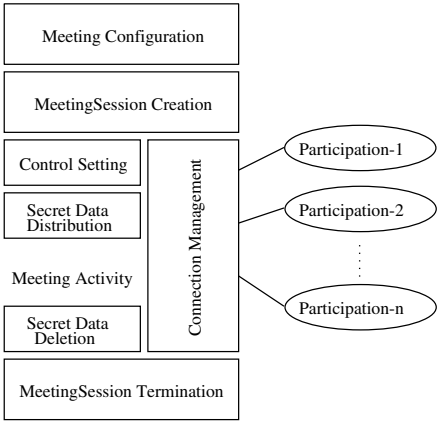


Fig. 4. Meeting Session

1. On the base of pre-determined configuration as above descriptions, the meeting manager creates a new meeting session. The meeting session will transit to one of the three security levels in Fig.5 with duration time, and role sets.
2. Then **Connection Manager** waits connections of participants (in our model, the clients). When there is a connection, RM examines user name and password to that client. If it is right, TM is created for that client.
3. The TM take charge of controlling the client directly. At first, TM send the initial control command to the client, for example as follows.

   ```
   device[floppy].controlstate=ReadOnly
   device[floppy].duration=2000/1/24/11:00-12:00
   application[browser].controlstate=executable
   application[browser].duration=2000/1/24/11:00-12:00
   ```

   The TM waits the answer to confirm the right operation from the client.
4. If the control state of the client is reliable, the TM sends secret data to the Device Manager on the client.
5. The participant now can join the **Meeting Activity** and read the secret data and communicate with another participants.

6. When the participant wants to get out, the TM sends the control command to the client for abolition of secret data basically, and confirms its state.
7. The TM then remains the log about the activity of client, then releases the control to the client in charge, and would be terminated.
8. After meeting, all TMs will go through the above steps from 6 to 7. Finally, the session would be terminated.

4.2 Constraints

In this meeting system, there are several constraints about meeting activity as the followings:

Security-level Constraints
The security level of meeting session is determined in session creation by meeting manager. When the level needs to be changed in the meeting, the new level must be considered seriously since the meeting has secret material. In general, a lower security level meeting just can be changed to more higher security level meeting. However, the reverse is not forbidden, because high-security data are possible to be exported in result of change of meeting security. Instead of it, meeting manager create a new meeting session.

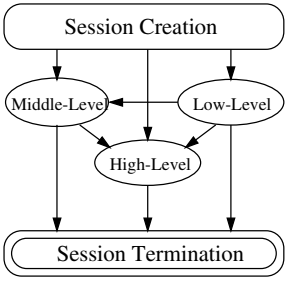


Fig. 5. Transition of Meeting Security Level

Time Constraints
The time allocated in role, terminal, device, application, and network, is the duration time in each other. The duration at device, application, network does not exceed of the terminal. The duration of terminal is also that of the role.

5 Conclusion

In this paper, the difficulties of opening the security meeting involving secret materials were discussed. The three conditions of the security meeting, that is,

restricted meeting place, authorized users, and secure data management, must be also maintained to the distributed environments. For this, we proposed a strictly restricted meeting system called **Secure Distributed Meeting System(SDMS)** where server controls all client resources such as applications, I/O interfaces, and network. The controls however are divided into multi-levels, low, middle and high security level. Low-security level gives no restriction for the client to use resources.

In reverse, the high-security level is for simulating the old centralized meeting that was held in a restricted meeting room. At the middle-security level, the meeting manager can make heterogeneous meeting that the resources can be configured by the request of the meeting. The processing from meeting creation to its destroy, and its constrains were also described in our SDMS model and Meeting Session descriptions.

The future work involves applications of SDMS model to another cooperative work, and adaptation to more unreliable distributed environments including mobile network.

References

1. John Baker, "Meeting manager", ACM interactions 3, 3 (May. 1996), Pages 42-43
2. Sandhu, R., Coyne, E., Feinstein, H.L., and Youman, C.E., "Role-based access control models", IEEE Computer 29, 2(Feb.), Pages 38-47
3. Ronald L. Rivest, Adi Shamir, and Leonard M.Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, Vol. 21, No. 2, Pages 120-126, 1978.
4. Kil-ho Shin, Masaki Kyohima, "Secure and Efficient Schemes to Entrust the Use of Private Keys", IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999.
5. Clark, D., and Wilson, D., "A comparision of commercial and military computer security policies", Proceedings of the 1987 IEEE Symposium on Security and Privacy, Pages 184-194.
6. Steven J. Greenwald, "A new security policy for distributed resource management and access control", Proceedings of the UCLA Conference on New Security paradigms workshops, 1996, Pages 74-86.
7. Kim C. Ng, Hiroshi Ishiguro, Mohan Trivedi and Takushi Sogo, Monitoring dynamically changing environments by ubiquitous vision system, in Proc. Workshop on Visual Surveillance, 1999.
8. Scott Oaks, "The Java Sandbox model", Java Security, O'Reilly, May 1998.

A Comparison of Two Architectures for Implementing Security and Privacy in Cyberspace

Reind van de Riet[1], Wouter Janssen[2], Martin Olivier[3], and Radu Serban[1]

[1] Department of Mathematics and Computer Science
Vrije Universiteit, Amsterdam
{vdriet,serbanr}@cs.vu.nl
[2] ERS-Deloitte&Touche, Amsterdam
wojanssen@deloitte.nl
[3] Department of Computer Science
Rand Afrikaans University
Johannesburg, South Africa
molivier@rkw.rau.ac.za

Abstract. In this paper we compare two approaches for implementing Security and Privacy systems in Cyberspace: a structured approach, such as done in Mokum, where access is governed by structure (of the classes), and two principles: the epistemic and the ontologic principle. The second approach is based on the use of capabilities, such as provided by ERP systems. **Keywords:** security and privacy, cyberspace, architecture, object-orientation, capability.

1 Introduction

Information about individuals is stored in many places which has led to the so-called privacy problem. In technical terms this means that access to this type of data is governed by specific rules specifying who is allowed/obliged to do what on this data under which circumstances. We will call these rules privacy rules.

When more than one company is involved in data about an individual, say a hospital H, and an insurance company IC, they may use Internet to communicate with each other using a secure connection based on crypto-techniques. Actually, there are actors in H and IC, represented by certain (software) agents whose functioning is prescribed in Work Flows (WF). The privacy rules will now directly refer to these WFs as they describe the circumstances in which the individual's information is needed. In another paper we will deal with the interaction of privacy rules and WF [TR2000]. In the current paper we will concentrate on two quite different architectures which must provide a trustworthy and flexible security systems. These architectures are: the ERP architecture, based on capabilities, and the Mokum architecture, based on structure.

The Mokum architecture has been developed in our group already some ten years ago. It is an object-oriented system where the objects are active sending

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 287–302, 2000.

each other messages and receiving answers. This is optimally suited to be represent by agents. To maintain privacy rules special objects can be created called keepers whose main task it is to maintain the integrity and Security&Privacy (S&P) rules for collections of objects. Using the static structure of the objects and special properties of these keepers it is possible to specify systems which are provably correct with respect to the privacy rules. This is based on two principles: the epistemic principle, which is implemented by reasoning features in the Mokum compiler and the ontologic principle which is provided by the Mokum run time system and checks the relationships between a caller and a callee. In [RB94] we have shown the universal applicability of these principles. In [RDR97] an extension of the Mokum system has been described which makes it possible to use Mokum also in a global fashion, that is in Cyberspace. In this paper we will give a sketch of the implementation details in Sect. 3. In [GRBO97] we have described how authorization tuples can be derived from a work flow specification and how these can be combined with a run-time system, also using the Mokum system. ERP systems integrate the work of employees in a company often defined in work flows with their usage of databases. In [RJG98] we have studied how security control in ERP systems is removed from the database system to a new layer in the ERP system itself. In Sect. 4 we will briefly describe this. In Sect. 2 we will introduce the hospital-insurance company example in more detail. In Sect. 5 we will make the comparison between the two approaches and give conclusions. In the area of architectures for Security & Privacy a lot of work has been done, and it is not possible in the framework of this paper to discuss that work here. We suffice with mentioning the Gendler & Gudes Model [GG97] which is a model in which the epistemic principle, as we will describe in Sect. 3, can be recognized, but not the ontologic principle. Also the MOOSE system described in [HTS97] comes close to the capability-driven approach we describe in Sect. 4.

2 The Example

Hospital H has medical staff and administrative staff. The medical staff consists of doctors and nurses (male and female). Patients are persons, having a medical record which is a private object kept in a collection of which the doctor of the patient (we assume there is only one) is the keeper. The nurse of the patient is not allowed to access the medical record, with the exception of an emergency and the doctor not being present. Also an employee HA in the administrative department has access to the financial records from which he can determine what a treatment has cost in terms of money; he sends invoices to the patients. He is keeping the administration in patient records in collections of objects of which he is the keeper. The Insurance company IC has several departments, one is the claim department, with claim employee CE, which treats claims issued by persons referring to a treatment by a doctor in H, recorded in the financial record. We have prepared three test queries and we will see in the next sections how they are dealt with in the two architectures.

1. Our first query is: a nurse wants to see the medical record of one of her patients. She is only allowed access to the medical record when the doctor of that patient is not present or when the patient is in a state of emergency.
2. The second query is: in order to check a claim, CE sends a message to the administrative person HA in H to ask the financial status of the patient. HA is entitled to give this information.
3. The third query is: in order to check a claim, CE wants to see the medical record of a client. He pretends being a nurse in an urgent situation and sends a request to the doctor who keeps this medical record.

In the diagram of Fig. 1 we have drawn the static structure of all objects involved. The diagram shows a lot of information necessary to deal with the queries. We see square boxes, denoting entities or object classes. The upper part contains the name of the class, the lower part the attributes. We have only given the name of the attribute, not the type of their values. Non-private attributes are indicated with a "+", while private ones are indicated with a "-". Almost all attributes are private, except "name" "address" and "doctor" (of patient). Is_a relations are indicated with an arrow with open head; arrows with a black head with label "k_o", indicate "keeper-of" relations. The latter relations are technically of interest for the Mokum implementation to be discussed in the next section; they are also of interest to the capability-driven approach as they indicate collections of objects, to which access has to be protected. These collections are:

- financial patients, with as private attribute: financial record
- medical patients, with as private attribute: medical record
- clients, with as private attribute: policy
- accredited persons, to be discussed below.

We also see the non-local aspects of the example: Three large boxes, indicated by dashed lines, one for the 'Civil Administration': CA, one for the 'Insurance Company': IC and one for the 'Hospital': H. Clearly, the claim employee CE and the Hospital administrator HA are in different sites, while a doctor D and nurse N are in the same site. The class employee can be considered to be defined in two sites: IC and H. Connecting certain classes to a location we does not mean that the objects of that class are necessarily stored at that site. That depends of the architecture chosen. We will discuss this site dependency in Sect. 3.2. For the Hospital administrator to know who is allowed to ask questions about financial records there is a collection of accredited persons, such as CE.

3 The Structural Approach in Mokum

Mokum is a system in which structure and behaviour of active objects can be defined. It is amply described in [DR94]. For this paper the following is relevant: All objects have one or more types and exist in collections. Each collection has a keeper, which is also an object, but with some responsibilities and some

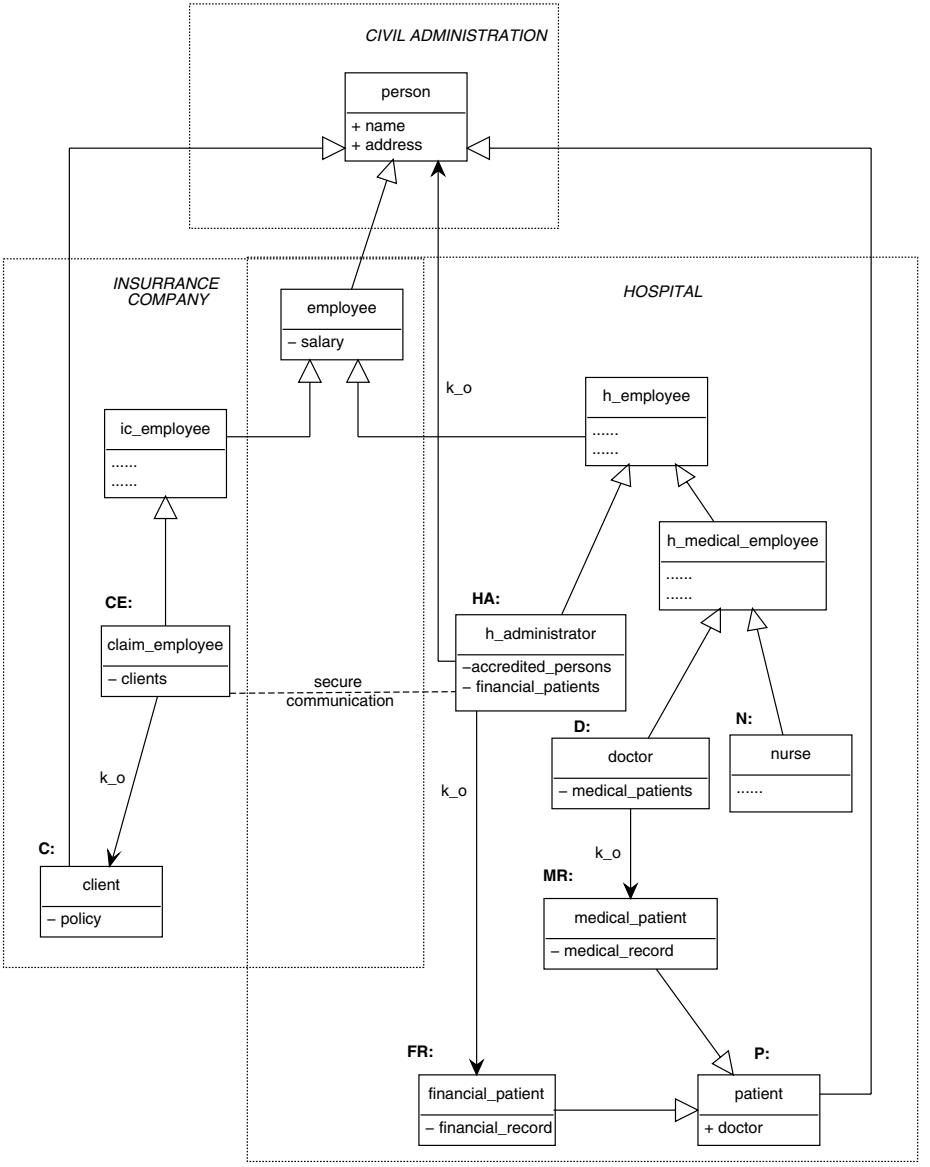


Fig. 1. The Structure of the Hospital-Insurance-Company Example

privileges. The responsibilities concern integrity and security constraints for their collections and the privileges concern the accessibility to the objects in their collections. To be a little bit more precise: a type specification defines the form and behaviour of the object in the usual way: a type has attributes, procedures, restrictions and a script. Attributes have themselves a type, namely the type of their values, usually called domain; they may be user-defined types or basic types. Attributes can be private, in which case they can be accessed in a controlled way, to be specified in a moment, and non-private, in which case they are accessible by any object. **Attribute values are the only things which can be defined private, and which are protected, in Mokum**. All other properties of an object: identifier, name, type, state, names of triggers are public and can be seen/used (but not changed) by any object. For this paper we only look at accessibility because of our emphasis on privacy. Therefore accessible means readable and non-accessible is non-readable. Private thus means non-readable. In another paper we will show that the results of this paper can also be applied to a system where read and write and even other actions are possible. The trick is to work on two levels.

Important for the structural properties of our system is the is_a relation: `type A is_a B`, with the well-known meaning that objects having type B have also all properties of objects type A. In the script part the behaviour of an object is specified. Syntactically, a script consists of several pieces of code (comparable with "methods" in C++) which define the actions of the object on an incoming message (trigger), sent by (another) object, or sent by the object itself as a timer message. Each piece of code is indicated by `at_trigger`. The object can be in different states. The code in the script can make use of the attributes defined in that type, and all inherited attributes, private and non-private. Furthermore, the attributes of objects being member of a collection, the current object of which is the keeper are also accessible: private and non-private. This is the main way to define a provably secure system. The proof depends on application of the simple rules given above. There are simple Prolog predicates to carry out this proof.

Accessibility is now defined by means of two principles, the epistemic and the ontologic principle:

1. The *epistemic principle* depends on visibility: within a piece of code which attributes are visible? This is a query which is answered by the Mokum compiler and boils down to a simple piece of Prolog code as the visibility depends on a simple reasoning mechanism (therefore "epistemic"), and is based on the following: Non-private attributes are visible in all type definitions; for private attributes, the type definition in which the attribute is defined has evidently the property that that attribute is visible. Moreover, an attribute is visible within a type definition when that attribute is visible in the definition of a super type or when that type is the type of a collection keeper, whose collection has elements with a type in which that attribute is visible. In Prolog, this is defined very precisely as follows:

```
vis(A,T):- is_a_attr(A), is_a_type(T), not private(A).
vis(A,T):- is_a_attr(A), is_a_type(T), private(A),
           (attr(T,A,_); (attr(T,_,[coll,S]); is_a(T,S)), vis(A,S)).
```

Applying this to a few objects, of the diagram in Fig. 1, of which the Prolog facts are given in the appendix, we see the following:

```
?- vis(medical_record,T).          ?- vis(financial_record,T).
     T = doctor ;                       T = h_administrator ;
     T = medical_patient ;              T = financial_patient ;
```

From these examples we immediately see that for our three queries:

(a) the occurrence of: `medical_record of P` in the script part of nurse is not allowed.
(b) the occurrence of: `financial_record of P` in the script part of the claim employee is not allowed.
(c) the occurrence of: `medical_record of P` in the script part of the claim employee is not allowed.

and vice versa: asking what is visible by objects of type claim_employee, and nurse,

```
?- vis(A, claim_employee).        ?- vis(A, nurse).
 A = name ;                        A = name ;
 A = address ;                     A = address ;
 A = salary ;                      A = salary ;
 A = doctor ;                      A = doctor ;
 A = clients ;
 A = policy ;
```

2. Evidently, the maintenance of the epistemic principle is not enough for a secure system, as one doctor can now access the medical records of patients of another doctor. Therefore, there is a second rule: the *ontologic principle*, or run-time check (on how things "are", therefore "ontologic"), which says: suppose the access involves an attribute of object O. The caller object CO must be the same as O, or must be one of the keepers of O. So, combining the two rules we get as rule for accessability: `acc(CO,O,A)`, with

```
acc(CO,O,A):- has_attr(O,A), not private(A).
acc(CO,O,A):- has_attr(O,A), has_type(CO,T), private(A),
              vis(A,T), (CO=O; keeper(CO,O)).
```

Assuming that only an insert operation of an object into a collection by its keeper is allowed, it is evident that maintaining this rule by Mokum's run time system is a guarantee that doctors cannot access other doctor's patient's medical records. In the appendix a few examples are given of objects of type doctor, nurse and patient, on which the accessibility predicate can be and has been applied.

We now give the scripts of the types of the objects involved, from which one can immediately see that sending, receiving, checking and answering a message is defined together. First, (a piece of) the script of a nurse, relevant for our security procedure:

```
at_trigger ask_for_medical_record:
        /* forward this request to the doctor of the patient */
```

```
        P from patient of message,
        D from doctor of P,
        send(D, nurse_asks_for_medical_record, message),
        next(active). /* meaning: take on active state*/
```

Next the script of a doctor's type; note that the nurse sends a special kind of message to a doctor:

```
at_trigger nurse_asks_for_medical_record:
    /* check that a nurse sent a medical record message. */
    N from sender of message,
    has_type(N, nurse),
    has_type(message, medical_record_message),
    P from patient of message,
    /* check that it is a patient of this doctor. */
    select (P1 in patients where P = P1),
    /* if doctor is absent and when the the patient is in a
       state of emergency then put the record in response */
    presence = 'false',  state of P = 'emergency',
    medical_record of P to response of message,
    next(active).
```

Finally, we show the script dealing with the request of the claim employee of the insurance company to the administrative hospital person:

```
at_trigger ask_for_financial_record:
    /* check sender of message is claim_employee */
    S from sender of message,
    /* check that sender of message is one of the "accredited"
       claim_persons known to this administrator */
    select(A in accredited_persons where A = S),
    /* check that client in message is one of the financial
       patients of this administrator */
    C from client of message,
    select (F in financial_patients where C = F),
    /* send financial record as response. */
       financial_record of C to response of message,
    next(active).
```

What happens when the claim employee pretends to be a nurse by performing `add_type(me, nurse)` and in this way tries to get the medical record by sending the doctor an appropriate message:

```
D from doctor of P,
send(D, nurse_asks_for_medical_record, message)
```

When it so happens that the doctor is absent and the patient is in a state of emergency, he gets the record, because types are not protected. We see that the test is not strong enough: instead of a test on the type, it should be a test on the sender being a member of a collection of nurses.

3.1 Connecting Real Persons with Their Alter-Egos

One aspect has not been dealt with: how get real persons contact with the system and get answers to their queries? The answer is that each individual has an object of type alter-ego_type, which as an object can send and receive messages to other objects. They can also communicate directly with the person they belong to through a special menu-driven protocol. The details are described in earlier papers, focusing more on the Alter-egos (see [RJ97]). As an example, a real doctor who wants to see a medical record of one of his patients, logs in, upon which his Alter-ego is activated; through a menu he chooses the task to be carried out. His Alter-ego then sends an appropriate message to the specific doctor object. In the script of the type of a doctor we find a trigger reacting on a message coming from an Alter-ego. The security check carried out here is nothing else than looking at the collection of doctors to see that this message was sent by the Alter-ego of a real doctor. The security checks for such accesses are very similar for all keepers of collections, so we don't discuss them here further.

3.2 Global Mokum in a Distributed Environment

Above we have assumed that there is one Mokum system, while in practice we have the situation that there may be many sites involved: one for the Civil Administration, CA, in which personal data is stored, one for the hospital H and one for the Insurance Company IC. We shall now see how a transition from one Mokum system at one single site to a configuration of several Mokum systems at several sites is a simple one. The transition will be such that the configuration of communicating Mokum systems is in its operation equivalent to one Mokum system, which is running at one site.

For our example of Sect. 2 we assume that, as indicated in Fig. 1, the personal data about a person P (or alter ego) is stored in an object with a type created in CA. Basic attributes such as name, address and birth-date are stored in CA. Information about insurance policies and claims is stored in IC and information about the medical situation is stored in H. As to how these data are connected, we assume that there is one object identifier for P. In fact when P was born, CA created that identifier (in a sense one can compare it with the social security number, in Dutch the SOFI number). When P gets a new type such as client in IC or patient in H, new tuples are assigned to P's object identifier: in IC to store the client's data and in H to store patient's data. We assume that CA, H and IC are Mokum systems, knowing each other, that means as Mokum programs they all three run as instances of one Mokum specification, namely the specification already given in Fig. 1. Later we will relax this condition. So in all three programs identifiers of types, procedures, messages and attributes are known. They differ in the way variables are initialized: in CA person objects have been created first, after which these persons become employees in IC and H and after that persons can become patients. How the synchronization of this is taken place is of no concern for us now. We simply assume that somewhere in the past these objects have been created and types have been assigned to these objects.

How is the access of attribute values taking place in this distributed environment? Assume `O.attr` is asked by a caller C. First assume that the attribute is public. In the Mokum system where C's call takes place the combination `(O, attr, val)` is looked up, if it is found the corresponding attribute value val is the one C is looking for. If it is not found the Mokum system does a broadcast to the other Mokum systems: "who has `(O, attr, val)`?" When some Mokum system responds positively, again the value is found. When no response is given `O.attr` has not got a value. In principle it is possible that more than one system is responding with even conflicting values. This points at the violation of an integrity constraint, namely the one which stipulates that an object can have a type only zero or one time. Therefore, as attribute names are unique (or are made unique using type names and site names), `O.attr`, if it exists, can only exist at one site.

Access protection is handled as follows: All Mokum systems have the same information about types, private attributes, is-a and keeper-of specifications. So for the epistemic rule the compilers of the three systems are doing the same analysis as the compiler of the one and single system. Therefore the outcome will be the same. As to the ontologic rule look at the situation that C is a keeper of a collection Coll. If O is a member of Coll access is allowed, otherwise it is not allowed; exactly the same as in the single system. It is not important where O is located. In fact there is no notion of O's site.

It is also interesting to see how a message is being sent. Suppose the object C is doing: `send(Object identifier, name of trigger,...)` where the name of trigger is, like the name of an attribute, unique to the type and script. So, as above, when this type is in the same Mokum system as C's call, the situation is the same as in a local Mokum system. When this is not the case a broadcast is done to the other sites: "who has the trigger of this Object"? In principle no more than one answer may come back. When there are more than one there is an integrity constraint, which should be prevented. When there is one, the message is sent in a secure way, to the responding system and that system acts on this message. The above description of a global Mokum system is a very simple extension of local Mokum's S&P implementation indeed.

For the example queries we see that there is no difference when they are issued in a local Mokum system or in a distributed one. It is of course necessary that in all three sites: CA, IC and H the same Mokum program MP is running. We can relax this condition a bit. Take IC: suppose the Mokum system which is actually used there is MIC, in CA it is MCA and in H it is MH. It should be possible to define the intersection and union operations on Mokum programs. In another study we look at this problem. Here we simply state that when the names used in two Mokum programs MA and MB are different, we simply can make a new Mokum program MP by concatenating MA and MB. It can be proved that the behaviour of MA and that of MB is equivalent to the behaviour of MP. If names are not different then they can be made different of course. In all three cases the actions to be carried out are very simple. We make a list of the assumptions to be sure that access is properly given:

1. Within the scripts the code for the events is properly chosen;
2. The structure of types and in particular the way the is-a and keeper-of relations are coded in the Prolog rules is done properly (which guarantees that the epistemic principle can be applied);
3. The insertion of objects in collections of appropriate keepers is done properly (which guarantees that the ontologic rule is maintained properly);
4. The code of the Mokum system to implement the access rules is done properly (the whole Mokum run-time system is some 20 pages, as is the compiler);
5. The way messages are sent to objects is done in a safe way.

3.3 An efficient Implementation of Global Mokum

To get the value of an attribute or to direct a message to its appropriate site can be cumbersome because the mechanism of broadcast is used. It is the price for flexibility. It is also possible to attach a site name to the type and thus to the attribute name and to the trigger name. No broadcast is then needed, but messages can immediately be sent to the proper site, and the same for attribute values. However, the uniqueness of attribute values needs to be maintained. In [RDR97] we described a prototype of global Mokum that deals with issues of remote message passing, object serialization and broadcasting. That implementation uses Java RM1 and a network of servers that handle object registration and locates remote objects. An extra security layer can be provided, that allows two parts of the same object to reside in several sites and communicate safely over the network. Problems which occur when objects are replicated, also for efficiency reasons, are momentarily being studied.

4 Capability-Driven S&P Systems

An ERP system is one single system, usually a product of one software company, such as SAP in Germany or its Dutch competitor Baan, both active world wide, the core of it is a (distributed) database and an integrated collection of services, in the form of application programs (usually written in a special language, such as ABAP for SAP), providing the workers in the company with the necessary services, to communicate with the databases and with each other. The services provided are all the services, in which IT is involved, in a company, or Enterprise, such as: financial, personnel, accepting and delivery of orders, planning of production, ordering products elsewhere, etc. These services usually fit well in formally described specifications of the tasks the employees are to fulfill, sometimes to such an extent that there exists a document describing these tasks in the form of Work Flow. It may even be that the Work Flow package includes a Work Flow engine which actually directs, by sending e-mail messages, the employees in their tasks.

By stressing the integration factor, vendors of ERP systems make it very tempting for the management of companies to install these systems, as they have seen the difficult problems raised by the non-integratedness of traditional

systems, usually designed by some enthusiastic employees in an IT department, which cannot be held responsible because they have left the company. Vendors of ERP systems make a very strong point by offering systems designed especially for certain branches of industry. Car industry is different from Food industry. The ERP vendor can afford to have IT specialists for these special branches. A consequence of this trend is that the structure of companies in similar branches of industry will become more alike.

4.1 Security in ERP Systems

When the ERP system is used in a company to deal with all IT problems in an integrated way, it is natural that defining and maintaining Security rules form an important part. The list contains: user authentication, separation of duties, need to know, just-in-time, role-versus-task, security administrator, logging. For a more complete description see [RJG98].

4.2 Capabilities for the Hospital-Insurance Company Example

In this subsection we will follow what a nurse N is doing to get access to the medical record of a certain patient P. After authentication N is identified by a user identifier, which points to her User Record UR. When N wants a certain transaction `T` to be executed, she chooses `T` from a menu, some data `D` and a key K, identifying the records from `D` on which `T` will be applied. Before `T` is executed, it is checked whether N possesses a capability where the combination `(T, D, K)` is in the list of allowed transactions. In our case `(read, medical records, P)`. In general, the capability may be value-dependent, where values in the capability can be compared with current data in the database. This data may be used when the capability is put in a user's profile, or it may be used in the transaction code itself as an additional (run-time) check. Capabilities can be combined in so-called profiles. A profile is generally speaking a list of capabilities. These lists can be assigned to groups of persons, or to persons acting in specific roles in order to be able to do their jobs. The nice thing with ERP systems is that the organizational structure of the personnel can be used integrally with giving out capabilities. There are three stages in the security process using capabilities to be looked at:

1. The *definition of a capability class* (in SAP these are called "authorization objects", many are pre-defined, but they can also be defined for specific purposes, such as the hospital). In general these classes have the following appearance:

   ```
   class auth_obj
      fctn: T
      data: D
      key: K
      sel: selection_criterion
   ```

   For the capability class for nurses we thus may have a class:

```
class auth_obj_nurse
   fctn: read
   data: medical_records
   key: patient_id
   sel: patient_id.state=emergency and not present(patient_id.doctor)
```

A capability class for doctor's capabilities looks as follows:

```
class auth_obj_doctor
   fctn: read and write
   data: medical_records
   key: patient_id
   sel: patient_id.doctor = <USER_ID>
```

Here we see that capabilities can be made adjusted to its owner: the user's id will be filled in at the moment of creation of a capability for a specific person, in our case a doctor.

2. *Creation of specific capabilities for employees* is the second phase of the security process. When an employee signs in, the authentication procedure assigns a set of capabilities to the user record of this employee taking into account the organizational hierarchy, as given in the personnel database. This is done automatically. This database reflects the diagram given in Fig. 1 very closely, regarding the personnel part. In the case of a nurse the capability can be a fixed one, but in the case of the doctor a special capability is created in which the doctor's identification is sealed in. So a doctor with `USER_ID` 1001 gets the capability attached to his user record:

```
capability auth_obj_doctor_1001
   fctn: read and write
   data: medical_records
   key: patient_id
   sel: patient_id.doctor = 1001
```

For nurses and other personnel similar capability objects are created, sometimes adapted to the employee, as doctors above, sometimes for a whole group, as the nurses.

3. The third stage of the security procedure is the *checking of the availability of a specific capability* when the actual database access takes place. The doctor D requests a certain medical record of a patient P, so he types in `(read, medical_records, P)`. The system, knowing D, looks at his user record and finds `auth_obj_doctor_1001` as capability. The function: "read" and data: "`medical_records`" is found OK, so a first access to `medical_records` is executed with the modified query:

```
SELECT patient's_anamnesis FROM medical_records
WHERE patient_id = P AND patient_id.doctor = 1001
```

When this query is successful we can be sure that the security constraint is maintained, so that the answer can be given to the doctor. If not, evidently either the patient with P as identifier does not exist, or this doctor is not P's doctor. Similarly, when a nurse N wants access to a medical record of a patient with identifier P, she also chooses: `(read, medical_records, P)`. Now the system knows N and her user record which contains `auth_obj_nurse`. The query now executed is:

```
SELECT patient's disease FROM medical_records
WHERE patient_id = P AND (patient_id.state = emergency
  AND not present(patient_id.doctor))
```

Also in this case we see a proper translation and execution of the security rules. In both cases we have seen the application of a very old principle: query modification to implement integrity and security rules. (see [St76])

Above we have seen the three stages for security access. At every stage the security administrator involved. For a local ERP system the situation is now clearly described.

4.3 Global Access Control in Capability-Driven Security Systems

The situation with cooperating systems in an ERP system, like SAP, can best be described using our IC-H (insurance company-hospital) example: when the claim employee CE wants to have regular access to the financial record of a patient in H an object CE' within the system of H is created as a virtual employee, with user record and capabilities. For this case the following capability class is defined:

```
class auth_obj_CE
   fctn: read
   data:financial_records
   key: patient_id
   sel: patient_id.insurance_company = IC
```

When CE wants to issue a query, he sends the query to CE' (properly encrypted of course). Then CE' actually poses the query and a security process is carried out exactly the same as for local employees. We see a very simple and clear solution for the global security problem.

5 Comparison of the Two Systems

In comparing the two approaches: structure-based and capability-driven, we look at the different kinds of accesses which have to be protected in an ERP system.

1. **Definition of capability classes;** SAP R/3 comes with about 600 different capability classes, which are needed in an average system. In addition to these classes special capability classes can be tailor-made.
2. **Making capabilities** for roles and groups of employees. In a typical company there may be 50 of these.
3. **Assigning capabilities** to these 50 roles and groups of employees.
4. **Checking the presence of capabilities in the application programs:** Typically there are some 200 information objects to be protected, on which some 20 different operations can be carried out, each one involving three capability classes on the average. An information object can be compared with a type in Mokum, such as financial_patient, medical_patient, etc. We

thus estimate that the number of attachments of capabilities to information objects is some 4000. The places where these attachments can be found are in the numerous application programs where they check that users have presented the proper capabilities. The checking itself is done in the next step.

5. **Checking the presence of capabilities at run-time** takes place.

Let us compare the above with the structure-based approach of Mokum:

1. **definition of capability classes:** No capability classes or similar things are present.
2. **Making capabilities:** No capabilities or similar things are present. In fact, the structure of the types as defined in a diagram like Fig. 1, together with the: epistemic and ontologic principles, define most of the access rules. Doctors accessing their own (medical) patients, or administrators treating the (financial) patients needs no special checking.
3. **Assigning capabilities**. In the Mokum approach this is not necessary.
4. **Checking the presence of capabilities in the application programs:** This is comparable to specifying the exceptions in the scripts of the keepers. So, the above mentioned 4000 places where capabilities have to be checked, can be compared with a number of different triggers (such as for nurse) which is an order of magnitude smaller.
5. **Checking the presence of capabilities at run-time:** No checking takes place, except for the automatic checking of the ontologic relations, as defined in Sect. 3.

Although mentioning the above numbers is rather risky, but the numbers give a feel for the complexity of the situation, and makes the comparison a bit more realistic. The comparison shows that the structure-based approach leads to a system which is an order of magnitude more trustworthy than the capability-driven approach. Another big difference between the structural approach and the capability-driven approach is that in the first one no capability maintenance is necessary. Considering that in a practical situation one may need thousands of capabilities, this is an advantage for the structural approach.

6 Conclusion

We have shown how in two architectures Security & Privacy rules are implemented and result in code, which have been compared in a general way, looking at Object-Orientation features, and at the complexity involved, using some numbers from practical experience. The main conclusion is that: The structure based approach, such as applied in the Mokum system, is better checkable with respect to S&P rules than the capability-driven approach as in the ERP system SAP R/3 and therefore more trustworthy.

References

[DR94] Dehne, F., R.P. van de Riet: A Guided Tour through Mokum 2.0, IR-368, Faculteit Wiskunde en Informatica, VU, October 1994.

[GG97] Gendler-Fishman, Masha, Ehud Gudes: Compile-time Flow analysis of Transactions and Methods in Object- Oriented Databases, in: T.Y. Lin, Sh. Qian (Eds.), Proceedings of Eleventh IFIP WG11.3 Working Conference on Database Security, Lake Tahoe, 1997, pp.88-10

[GRBO97] Gudes, Ehud, Reind van de Riet, Hans Burg, Martin Olivier: Alter-egos and Roles Supporting WorkFlow Security in Cyberspace, in: T.Y. Lin, Sh. Qian (Eds.), Proceedings of Eleventh IFIP WG11.3 Working Conference on Database Security, Lake Tahoe, 1997, pp.152-166.

[HTS97] Hale, John, Jody Threet, Sujeet Shenoi: Capability-Based Primitives for Access Control in Object-Oriented Systems, in: T.Y. Lin, Sh. Qian (Eds.), Proceedings of Eleventh IFIP WG11.3 Working Conference on Database Security, Lake Tahoe, 1997, pp.88-103.

[OGR98] Olivier, M.S. E. Gudes, R.P. van de Riet, J.F.M.Burg: Specifying Application-lev el Security in Workflow Systems in: R. Wagner (Ed.) Database and Expert Systems Applications, IEEE Computer Society, 1998, pp. 346-354.

[RB94] Riet,R.P.van de, Beukering, J.: The Integration of Security and Integrity Constraints in MOKUM in: J.Biskup, M.Morgenstern, C.Landwehr (Eds), Proceedings of IFIP WG11.3 Working Conference on Database Security,IFIP/North Holland,1994,pp. 223-246.

[RB96] Riet, R.P. van de, J.F.M.Burg, Modelling Alter Egos in Cyberspace: Who is Responsible? in: Proceedings WebNet96, San Francisco, AACE (Association for the Advancement of Computing in Education), Charlottesville, USA pp. 462-467,1996.

[RB97] Riet, R.P. van de, J.F.M.Burg, Modelling Alter Egos in Cyberspace: using a Work Flow management tool: who takes care of the Security and Privacy in: S.Lobodzinsky, I.Tomek (Eds.), Proceedings of WebNet97, Toronto, Association for the Advancement of Computing.

[RBK91] Rabitti, F., E. Bertino, W.Kim, D.Woelk: A Model of Authorization for next-generation database systems, ACM Transactions on Database Systems, Vol. 16, No. 1, pp. 88-131, 1991.

[RDR97] Radu,S., F.Dehne and R.P. van de Riet, A first step towards distributed Mokum, Technical Report 428, Computer Science Department, Vrije Universiteit, 1997.

[RG96] Riet, R.P. van de, E. Gudes: An Object-Oriented Database Architecture for Providing High-Level Security in Cyberspace in: P.Samarati, R. Sandhu (Eds.), Proceedings of Tenth IFIP WG11.3 Working Conference on Database Security, Como 1996, pp.92-115.

[RJ97] Riet, R.P. van de, Andrea Junk & E. Gudes: Security in Cyberspace: a Knowledge- base Approach, Data and Knowledge Engineering, Vol 24, Nr.1, North Holland, 1997, pp. 69-98.

[RJG98] Riet, R.P.van de , W.Janssen, P.de Gruijter: Security moving from Database Systems to ERP Systems in: R. Wagner (Ed.) Database and Expert Systems Applications, IEEE Computer Society, 1998, pp. 273-280.

[St76] Stonebraker, M., E.Wong, P.Kreps, G.Held: The Design and Implementation of INGRES, Transactions on Database Systems, ACM, Vol 1, Nr. 3. pp. 189-222., 1976.

[TR2000] Wouter Teepe, Reind van de Riet, Martin Olivier: WorkFlow Analyzed for Security and Privacy in using Databases; submitted for publication. 2000.

[Va95] Varadharajan, V. Distributed Object System Security, *Information Security - the next Decade,* Edited by H.P Eloff and S. H. von Solms, Chapman & Hall, 1995, pp. 305-321.

A Appendix: the Example in Prolog Terms

For the Hospital-Insurance-Company example we have a list of the following facts, written in Prolog, as a direct translation of the diagram in Fig 1.:

```
type_set([person, employee, ic_employee, h_employee, claim_employee,
    client, h_administrator, h_medical_employee, doctor, nurse,
    medical_patient, patient, financial_patient]).
attr_set([name, address, salary, clients, policy, financial_patients,
  accredited_persons, patients, financial_record,
  medical_record, doctor]).
private(salary).                      private(clients).
private(policy).                      private(financial_patients).
private(accredited_persons).          private(patients).
private(medical_record).              private(financial_record).
is_a(employee, person).               is_a(ic_employee, employee).
is_a(h_employee, employee).           is_a(claim_employee, ic_employee).
is_a(client, person).                 is_a(h_administrator, h_employee).
is_a(h_medical_employee, h_employee). is_a(doctor, h_medical_employee).
is_a(nurse, h_medical_employee).      is_a(patient, person).
is_a(medical_patient, patient).       is_a(financial_patient, patient).
attr(person,  name, [simple,int]).
attr(person,  address, [simple,string]).
attr(employee,  salary, [simple,real]).
attr(claim_employee, clients, [coll, client]).
attr(client, policy, [simple, string]).
attr(h_administrator, financial_patients, [coll, financial_patient]).
attr(h_administrator, accredited_persons, [coll, person]).
attr(financial_patient, financial_record, [simple, string]).
attr(doctor, patients, [coll, medical_patient]).
attr(medical_patient, medical_record, [simple, string]).
attr(patient, doctor, [simple, string]).
```

To make a simple simulation possible, some facts have been added about doctors john and claus, nurse mary and patients pete and ann. Also some rules have been added, which, with the rules about accessibility in Sect. 3, complete the simulation tools.

```
has_type(john, doctor).               has_type(claus, doctor).
has_type(mary, nurse).                has_type(pete, medical_patient).
has_type(ann, medical_patient).       value_of(patients, john, [pete]).
value_of(patients, claus, [ann]).

has_type(O, T) :- is_a(S, T), has_type(O, S).
keeper(CO, O):- has_type(CO, T), has_type(O, S), attr(T, AC, [coll, S]),
                member_of(O, CO, AC).
is_a_attr(A):- attr_set(AS), member(A, AS).
is_a_type(T):- type_set(TS), member(T, TS).
has_attr(O, A):- has_type(O, T), attr(T, A,_).
member_of(E, CO, A):- value_of(A, CO, V), member(E, V).
```

Organizations and Collective Obligations

Lambèr Royakkers and Frank Dignum

Eindhoven University of Technology
Dept. of Technology Management and Dept. of Mathematics and Computer Science
P.O.box 513, 5600 MB Eindhoven, The Netherlands
L.M.M.Royakkers@tm.tue.nl, dignum@win.tue.nl

Abstract. Multi-Agent Systems are computational systems in which a collection of autonomous agents interact to achieve a certain task, for example to fulfil an obligation directed to the whole group, i.e., a collective obligation. Since, such a collective obligation is beyond the capacity of an individual agent, the agents have to communicate, cooperate, coordinate and negotiate with each other, to achieve the collective task: the fulfilment of the obligation. In this paper we discuss and formalise collective aspects of obligations and commitments. Collective obligations are analysed and formalised in a deontic logic framework. The notions of individual and collective commitment are defined to specify which individual has the responsibility to fulfill an 'internal' obligation as part of the collective obligation. In distributed artificial intelligence (DAI) theories of organisations, it is emphasized that 'commitment' is a crucial notion to analyse a collective activity or the structure of an organisation. In this paper we give a first attempt to formalise the notion of commitment to determine which plan has to be followed to achieve a joint goal, i.e. the fulfillment of a collective obligation by using several concepts as commitment, delegation and authority-relation.

1 Introduction

The aim of this work is to introduce some notions of commitment and group structure by delegation to determine the relation between individual agents and their collective obligation to accomplish some goal. A collective obligation is an obligation aimed at a group of agents (addressees); i.e., the group as a whole is obliged to achieve a certain task. E.g. a program committee may have the collective obligation to review all submitted papers before a certain time. We are interested to explore how this collective obligation translates into individual obligations for the program committee members, e.g. review two or three papers, and the extra obligations for the program chair to divide the papers and monitor the process and make final decisions. In, e.g., [1] and [17], the collective obligation is formalized in a framework of deontic logic, which gives the opportunity to express which group of agents has the responsibility to bring about a certain situation (to express group liability, e.g. liability for a trading partnership) and to express the relation between the agents of a group. Such an approach is necessary for a good definition of our concepts and for developing a formal theory

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 302–311, 2000.

of groups, organizations, and collective actions, since there is *no Organization without Obligations.*

However, in these theories the groups are considered as separate 'institutionalized agents' as in the theory of relativised deontic modalities concerning norms for an individual, since now every group stands on its own, like agents. Such a group will be considered as a communicative, cooperative and coordinated group. A consequence of this interpretation is that we do not gain an understanding of the collectivity of the obligation concerning the relation amongst the individuals and the relation between the individuals (or subgroups) and their collective obligation to accomplish some goal. The transformation of collective obligation to the members happens automatically, by magic. The notion of commitment is lacking here as a *mediator* of such transformation (see [3]) to express issues like delegation, adaption, intention, responsibility, etc., which constitutes the theory of collective action in a narrower sense. This is one of the complex and interesting issues in the logical theory of collective agency.

Commitment links the agents with the joint goal (i.e. the fulfilment of the collective obligation), so that we can express to whom such an agent is committed to and to what that agent is committed to (see [2, 5, 15]). The core of such a commitment is delegation. Delegation is a basic ingredient for joint intentions, true cooperation and team work. It can be used for a plan-based definition of tasks, which have to be achieved by the agents of the group. In fact, in organization theories of DAI ([6]), negotiation systems and cooperative software agents, it is emphasized that 'commitment' is a crucial notion to analyse a collective activity or the structure of an organisation, through the allocation of some task by a given agent to another agent. This is basically correct, although not general enough: it is already restricted to the most complex (less basic) forms of commitment and delegation. We will start from more basic notions. With such a kind of analysis we gain an insight into different kinds of organization, and consequently for different kinds of commitment, and then for different ways to solve conflicts, both intra- and inter- agents. For instance, how to intervene when one of the agents decides to abandon the group or the common plan, or omits his task.

In this paper, we will still remain very generic, we do not deal with individual motivations, restrictions on the groups of agents, etc. It has to be seen as a first attempt to combine some notions (collective obligation, commitment and delegation) to formalise the individual responsibility to achieve a joint task, which gives new expressive power and is, therefore, subject to new intuitions.

This paper is structured along the following lines. In section 2, we discuss the collective obligation. Section 3 presents several notions of commitment. The collective commitment depends on the underlying structure of the group which will be discussed is section 4. Furthermore a first attemp is given to formalize the relation between the collective obligation and individual ones by using the previous notions. We finish with some conclusions.

2 Collective obligations

Although obligations most often are aimed at individuals, like in the statutes of law, obligations can also be aimed at a group. (since there are cases in which the 'addressee' is not a single agent, but a group). It is important in this context not to confuse a collective obligation for a group with a restricted general obligation. A restricted general obligation indicates that for every person in a group the obligation holds. E.g., cyclists have to give way to motorvehicles. That a restricted generalized obligation is different from a collective obligation can be seen easily from the following example, taken from Rescher ([16]):

'John and Paul are obliged that the table is moved across the room.'

If John alone accomplishes that the table is moved across the room, then the group (John and Paul) satisfies the norm, i.e., the obligation that the table is moved across the room. It does not follow from the obligation that Paul would have to help John. This example expresses that from a collective obligation no obligations follow automatically for the members of the group. For example 'John is obliged that the table is moved across the room' does not follow. Thus, the obligation does not require that everyone (in this case, John and Paul) in the group takes part in the act to accomplish the obligation, but it requires that the group accomplishes something (i.e., that the table is moved). We express this collective obligation in deontic logic as follows: $O_X(p)$, where 'X' refers to the set of 'boys', and 'p' to the statement 'the table is set'. In general, $O_X(p)$ can be read as 'it is obligatory for X that p'. 'X' is used for the expression of a *collective agent* or group. Again, collective obligation $O_X(p)$ does not mean that this is a restricted general obligation, i.e., that for every agent in X it is obligatory that p, but that X *as a group* has to accomplish that p.[1] If we want to emphasise that i and j are both 'essential' group-components to fulfill the obligation $O(p)$, then we can formalise this as follows:

$$O_{\{i,j\}}(p) \wedge \neg O_{\{i\}}(p) \wedge \neg O_{\{j\}}(p) \tag{1}$$

We can regard (1) as a definition of genuine group obligation or *cooperative obligation* (cf. [11]).

In our interpretation of $O_X(p)$, we consider the groups to be separate 'agents' as in the theory of relativised deontic modalities concerning norms for an individual (e.g. [9–12]), since now every group stands on its own, like agents. Such a group will be considered as a communicative, cooperative and coordinated group. A consequence of this interpretation is that we do not gain an understanding of the collectivity of the obligation concerning the relation amongst the individuals and the relation between the individuals (or subgroups) and their collective obligation to accomplish some goal. For that purpose we will introduce the notion of commitment.

[1] For the semantics of the collective obligation we refer to [17].

3 Commitment

Intuitively, the relation between the members of the group and their collective obligation is shaped by the commitment of the group to fulfill the obligation according to some plan and structure of the group. In this section we discuss some notions of commitment (as a descriptive ontology) which are crucial for the understanding of the relation between the individual agents and their collective obligation to accomplish some goal. Castelfranchi claimed in [2] that a notion of commitment is needed as a *mediation* between the individuals and the collective one:

> 'Commitment' is seen as the glue of the group, of collective activity: it links the agent with the joint goal and the common solution, it links the members' actions with the collective plan, it links the members with each other.

From the above description of commitment, we infer three notions we will use for the analysis of the collective obligation: joint goal, collective plan, and authority-relation. Before we will discuss these three notions, it is necessary to have a better understanding of commitment. Suppose that i and j agree that j will make dinner and i will do the dishes. Then each of them has an obligation to conform to the agreement, at least in the absence of countervailing factors. What is the kind of obligation in the agreement? The distinction we make is between *obligations of social commitment* and other kinds of so-called obligations. We distinguish three kinds of commitment: personal commitment, social commitment, and collective commitment. Before explaining what social and collective commitment is, it will be useful to characterize what we shall call 'personal commitment'.

3.1 Personal commitment

An agent is subject to a personal commitment if and only if the agent is the sole author of a commitment, and has the authority unilaterally to rescind it (cf. [8]). According to Cohen and Levesque ([13]) a personal commitment refers to a relation between an agent and an action. The agent has decided to do something, the agent is determined to perform an action (at the scheluded time), the goal (itention) is a persistent one. The way to capture such a persistence is to establish that the intention is abandoned only if and when the agent believes that the goal has been reached, or that it is impossible to achieve, or that it is no longer motivated. So, the agent can rid himself of it, only by changing his mind: the agent *has the authority* unilaterally to rescind his own decision.

Further, we assume that to commit to an action necessarily implies committing to some result of that action. Conversely, to commit to a goal always implies the commitment of at least one action that produces such a goal as result. Thus, we consider the action/goal pair $\tau = (\alpha, g)$ as the real object of commitment, and we will call it 'task'. Then by means of τ, we will refer to the action α, to its resulting goal g, or to both.

Many of the goals one might have, require the participation of two or more people for their achievement. Consider a simple transaction: If an agent wants

to buy something, for instance, he will require the help of someone who sells it. Therefore, we need the notion of social commitment.

3.2 Social commitment

A social commitment is a commitment between two agents: the commitment of one agent to another (cf. [20]).[2] The social commitment cannot be composed of two personal commitments of two agents, it is created by two agents together. It expressess a relation between two agents, more precisely, social commitment is a 3-argument relation:

$$\text{S-Comm}(i, j, \text{Achieve}_i(\tau)), \tag{2}$$

meaning that i (the committed agent) is committed to j (the agent to whom i is committed) to achieve τ. In this case, only agent j is in the position (has the authority unilaterally) to rescind it. Thus, agent j can rely on the persistence of agent i's commitment to achieve τ.

A necessary condition of a social commitment is the mutual knowledge by the two agents about the commitment: there must be an agreement. In daily life, this is done by expressions in conditions of common knowledge (if it is 'out in the open') as far as the two agents are concerned (cf. [7]), and not, for instance, by a contract. In [2], Castelfranchi shows that this condition is not sufficient. Two other conditions must be added:

1. The intention of agent i to achieve τ is a goal of agent j. Formally this can be expressed as

$$\text{S-Comm}(i, j, \text{Achieve}_i(\tau)) \rightarrow \text{Goal}_j(\text{Achieve}_i(\tau)). \tag{3}$$

   In fact, if i is socially committed to j to achieve τ, then also j is socially committed to i. This last commitment contains the *acceptance* of j that i achieves τ. Without this (implicit or explicit) acceptance, there is no social commitment, at the most a personal commitmment. E.g., if i promises j to knock him out, and j does not accept this, then there is no social commitment between these two persons, let alone a directed obligation of i towards j to knock j out. The intended action of i does not lead to a goal which j has in mind. Thus,

$$\text{S-Comm}(i, j, \text{Achieve}_i(\tau)) \rightarrow \text{S-Comm}(j, i, \text{Accept}_j(\text{Achieve}_i(\tau))). \tag{4}$$

2. Agent j is entitled (to control, exact, protest) by agent i to τ, if i is socially committed to j.

[2] In [2], the social commitment is considered as a relation between two or three agents. The third agent is the *witness*, who has a very crucial role in normative contexts (norms efficacy) (cf. [4]) and in contractual contexts implicating free riders and cheaters. For convenience, we will not discuss the third agent and therefore we will omit this from the paper.

This entitlement is not exact a permission, but a relation of 'entitlement' between the agents i and j means that j has the rights of controlling (exacting, protesting) the subtask of i. For convenience, we express this 'entitlement' with a directed permission:

$$\text{S-Comm}(i, j, \text{Achieve}_i(\tau)) \rightarrow \text{Permitted}^i_j(\text{Control}(\tau)). \tag{5}$$

According to Castelfranchi, a social commitment implies an obligation:

$$\text{S-Comm}(i, j, \text{Achieve}_i(\tau)) \rightarrow O_i(\text{Achieve}(\tau)). \tag{6}$$

The mistake of confusing commitment with obligation arises directly out of a conflation of the two distinct senses of the term commitment: the formula S-Comm(i, j, τ) only involves (unambiguously) the intensional sense of commitment (agent i *undertakes* an obligation to achieve τ from j), and the formula $O_i(\tau)$ the extensional sense of commitment (agent i has an obligation to achieve τ). So, in fact, S-Comm(i, j, τ) need not imply $O_i(\text{Achieve}(\tau))$; we can undertake obligations that we do not have (cf. [21]). Therefore, we think that (6) is too strong. A social commitment can not imply a juridicial norm, but at most a *directed* norm (which we will use):

$$\text{S-Comm}(i, j, \text{Achieve}_i\tau) \rightarrow O^j_i(\text{Achieve}(\tau)). \tag{7}$$

The directed obligation $O^j_i(\text{Achieve}(\tau))$ is read as 'agent i (the addressee) has an obligation towards j (the counterparty) that τ will be achieved.

3.3 Collective commitment

The collective commitment is defined as the internal commitment of a group (a collective agent) X given the underlying structure of the group with respect to the achievement of τ. So we represent the collective commitment as a conditional one:

$$\text{C-Comm}(X, \text{Achieve}_X(\tau)|\text{Plan}, \text{Struct(X)}), \tag{8}$$

meaning that the group X has a collective commitment to achieve τ given the underlying structure of the group or organization X. The underlying structure we discuss in the next section. A set of agents is internally committed to a certain intention under the condition that there is a mutual knowledge between the agents about the intention according to the structure of the group. In the following section we try to clarify the relationship between the social and collective one. This will help us to indicate who has the authority to rescind unilatterally some particular commitments in the collective one and who has the responsibility to a certain subtask. This also depends on delegation, which is the core of the social commitment relationship (cf. [3]).

A collective commitment presupposes an associated collective intention with respect to the goal and a common belief in the whole group that the plan is adopted by the committed members of the group. In this paper we do not give a further characterization of this collective intention, because it is not needed for our theory. We refer interested readers to [5].

4 The underlying structure

The collective commitment depends on the underlying structure of the group or organization. We represent the underlying structure as a 4-argument relation: $(X, T, \mathrm{Auth}(X,T), \mathrm{Plan})$.

The set X: The set X represents a non-empty finite set of agents, i.e. the agents of the organization: $X = \{i_1, \ldots, i_n\}$.

The set T: The set T represents the set of all individual sets of tasks. An individual set T_i of tasks contains the tasks of individual i belonging to his job description or his task repetoire. So, the set T can be expressed as

$$T = \{T_{i_l} | l \in \{1, \ldots n\}. \tag{9}$$

The authority-relation Auth: The possibility of delegation depends on te authority-relation within the organization/group. The formula $auth(i, j, \tau)$ with $\tau \in T_{i,j}$ indicates that agent i can delegate task τ to agent j. $T_{i,j}$ indicates the set of tasks of the job describtion of agent j about which agent i has authority (control), meaning that these tasks also are elements of the individual task-set of j. So $T_{i,j} \subset T_i \cap T_j$. The above member of the program committee can delegate his task to review some papers to his assistent, since he has some authority to delegate his task to his assistent. The same holds for the relation between the chairman of the program committee and this member.

Informally, in delegation an agent i needs or desires an action of another agent j and includes it in its own plan, i is trying to achieve some of its goals through j's actions; thus i has the goal that j performs a given action. We introduce an operator of delegation with three parameters: $\mathrm{Del}(i, j, \tau)$, meaning that i delegates the task τ to j. In this paper, we are only interested in the interaction between the delegating agent (client) and the delegated one (contractor) based on explicit agreement, which is called *contract* ([3]). In a contract, the delegated agent knows that the delegating agent is relying on it and accepts the task. If the agreement is settled, the delegated agent j is socially committed to the delegating agent i to achieve the agreed task τ for i. Formally:

$$\mathrm{Del}(i, j, \tau) \rightarrow \text{S-Comm}(i, j, \mathrm{Achieve}_i(\tau)), \text{with } \tau \in T_{i,j} \tag{10}$$

meaning that i is socially committed to agent j to achieve τ, if j delegates task τ to i. With (6), we can derive then

$$\mathrm{Del}(i, j, \tau) \rightarrow O_i^j(\mathrm{Achieve}(\tau)). \tag{11}$$

In a close contract the delegating agent i and delegated agent j collaborate in the collective plan that they share and know fully, in which the task is completely specified. In such a contract also the actions are completely specified how to achieve the task that i expects from j.[3]

[3] However, minimally specified (or open) delegation is fundamental, because it is also due to the delegated agents's ignorance about the world and its dynamics. In fact,

We state that $auth(i, j, \tau)$ holds if and only if i is permitted to delegate a task τ in the set $T_{i,j}$ to agent j:

$$\text{Permit}_i(\text{Del}(i, j, \tau))) \equiv auth(i, j, \tau), \text{with } \tau \in T_{i,j} \tag{12}$$

Usually the authority-relation is reflected in organigrams of companies or groups. The authority-relation is not symmetrical, and not transitive. We will represent the authority-relation of the group X by $\text{Auth}(X, T)$:

$$\text{Auth}(X, T) := \{auth(i, j, \tau) | i, j \in X, \tau \in T_{i,j}\}, \tag{13}$$

meaning that $\text{Auth}(X, T)$ is the set of all existing authority-relations between two agents in the set X of agents.

Collective Plan: To achieve a collective task τ, the organisation has to decompose the task τ (to achieve τ) into a number of subtasks over the agents in management positions. For example, if the program committee has the task to notify the authors of the submitted papers of acceptance before a certain deadline, this task can only be achieved if the work that has to be done, is shared out in several tasks over the members of the program committee. Therefore, the organisation needs a plan: a concrete manner to achieve the collective task. A plan includes *task division* and *task allocation* (see [5]). Task division is the decomposition of a complex task τ into subtasks $\tau_1, \ldots, \tau_n$. Task allocation indicates which agent of the group has to achieve which subtask of the complex task. This results in a set of individual subtasks that realizes the complex task (the collective goal). We represent a plan as $\text{Plan}(< i_1 : \tau_1, \ldots, i_n : \tau_n >, \tau, X)$, with

- $< i_1 : \tau_1, \ldots, i_n : \tau_n >$ the set of individual subtasks $i_l : \tau_l$ with $1 \leq l \leq n$,
- $\tau_l \in T_{i_l}$ and $i_l \in X$ for $l \in \{1, \ldots, n\}$,
- τ the collective task, with $\tau_1 \wedge \ldots \wedge \tau_k \rightarrow_X \tau$.

A plan determines the individual subtasks of the agents on the management positions. We note that we do not account yet for the fact that some goals might depend on other ones. So, a plan must also determine the *order* of subtasks: the temporal structure of the individual subtasks. For example, the notification of acceptance of a certain paper can only be done if it is reviewed by the delegated members of the program committee. Thus, an individual task can sometimes only be achieved if other tasks are already achieved. So, a plan does not only contain a set of individual subtasks, but also contains an order of the individual subtasks, i.e., a sequential composition of the individual subtasks.

From the collective commitment $\text{C-Comm}(X, \text{Achieve}_X(\tau) | \text{Struct}(X))$ with the following underlying structure $(X, T, \text{Auth}(X, T), \text{Plan}(< i_1 : \tau_1, \ldots, i_n : \tau_n >, \tau, X))$, we can derive the following social commitments:

$$\text{S-Comm}(i_l, X, \text{Achieve}_{i_1} \tau_l), \text{for all } l \in \{1, \ldots, n\} \tag{14}$$

frequently enough it is not possible or convenient to fully specify the task because some local and updated knowledge is needed in order for the part of the plan to be succesfully executed.

So, individual i_1 is socially committed towards the group X to achieve task τ_1, and by (7) also obligated towards group X: $O^X_{i_1}(\text{Achieve}(\tau_1))$. Futhermore, it depends on the authority relation $\text{Auth}(X,T)$ whether an individual can delegate his task. E.g., if $auth(i_1, i_2, \tau_1)$ holds, then agent i_1 can delegate task τ_1 to agent i_2. If he delegates the task, then agent i_2 is socially committed towards i_1.

With the notion of commitment we are able to represent the relation between the collective obligation and the individual directed obligations (following from the collective obligations):

$$O_X(\text{Achieve}(\tau)) \wedge \text{C-Comm}(X, \text{Achieve}_X(\tau)|\text{Struct}(X)) \rightarrow$$

$$O^X_{i_1}(\text{Achieve}(\tau_1)) \wedge \ldots \wedge O^X_{i_n}(\text{Achieve}(\tau_n)). \tag{15}$$

5 Conclusions

First of all, there are complex and interesting questions in the logical theory of collective obligation:

> From the point of view of methodological individualism, the action of a collective is in some sense composed of or determined by individual actions performed by the members of the collective. The general study of the nature of that composition, of the dependence of collective actions on individual ones, could be said to constitute the theory of collective action in a narrower sense. ([14])

A first attempt is made to 'constitute the theory of collective action in a narrower sense' by the introduction of commitment and to define a goal of a collective as a set of individual subtasks. The notion of commitment is very useful to determine which *plan* will be followed to fulfill a collective obligation (collective commitment) and to determine which agent has to perform (or is responsible to perform) a certain subtask - being part of the committed sequence - to achieve the joint goal (individual commitment). To indicate the responsibility of the individual agents to achieve their subtasks, we need the notion of commitment. As we already said, the fulfillment of a collective obligation can only be reached if the group decomposes their goal into a number of individual subtasks by *commitment*. We cannot appropriately influence these individual obligations if we do not exactly know the individual commitments of the group. We have shown how the relation can be made between the collective obligation and the individual directed obligations, through the use of underlying structure of the organization.

Finally, we note that 'right' and 'duty' play a role in the combination of collective obligation and commitment. This and other subjects, such as open delegation and maintenance of commitment are issues for future research. We gave a first step to realise the formalisation of the individual responsibility of a collective obligation.

References

1. Carmo, J. and O. Pacheco, Deontic and action logics for collective agency and roles, in: R. Demolombe and R. Hilpinen (eds.), *Proceedings of the Fifth International Workshop on Deontic Logic in Computer Science*, Toulouse, pp. 93-124, 2000.
2. Castelfranchi, C., Commitments: From Individual Intentions to Groups and Organizations, in: V. Lesser (editor), *Proceedings First International Conference on Multi-Agent Systems*, AAAI-Press and MIT Press, San Francisco, pp. 41-48, 1995.
3. Castelfranchi, C. and R. Falcone, Towards a theory of delegation for agent-based systems, *Robotics and Autonomous Systems* 24, pp. 141-157, 1998.
4. Conte, R., and C. Castelfranchi, *Cognitive and Social Action*, UCL Press, London, 1995.
5. Dunin-Kęplicz, B., and R. Verbrugge, Collective Commitments, in: M. Tokora (editor), *Proceedings Second International Conference on Multi-Agent Systems*, California, AAAI-Press, pp. 56-63, 1996.
6. Gasser, L., Social conceptions of knowledge and action: DAI foundations and open systems semantics, *Artificial Intelligence* 47, pp. 107-138, 1991.
7. Gilbert, M., *On Social Facts*, Princeton, 1992.
8. Gilbert, M., Obligation and joint commitment, *Utilitas* 11, no. 2, pp. , 1999.
9. Hansson, B., Deontic Logic and Different Levels of Generality, *Theoria* 36, pp. 241-248, 1970.
10. Herrestad, H. and C. Krogh, Deontic Logic Relativised to Bearers and Counterparties, in: J. Bing and O. Torvund (editors), *Anniversary Anthology in Computer and Law*, COMPLEX - TANO, pp. 453-522, 1995.
11. Hilpinen, R., On the Semantics of Personal Directives, *Ajatus* 35, pp. 140-157, 1973.
12. Kordig, C.R., Relativised Deontic Modalities, in: A.R. Anderson et al (editors), *The Logical Enterprise*, Yale University Press, New Haven, pp. 221-257, 1975.
13. Levesque, J.H., P.R. Cohen and J.H. Nunes, On acting tgether, in: *Proceedings of AAAI-90*, Menlo Park, California, American Association for Artificial Intelligence Inc, 1990.
14. Pörn, I., *The Logic of Power*, Basil Blackwell, Oxford, 1970.
15. Rao, A., and M. Georgeff, Modelling rational agents within a BDI-architecture, in: R. Fikes and E. Sandewall (editors), *Proceedings of the Second Conference on Knowledge Representation and Reasoning*, Morgan Kaufman, pp. 473-484, 1991.
16. Rescher, N., *The Logic of Commands*, Dover, New York, 1966.
17. Royakkers, L.M.M., *Extending deontic logic for the formalisation of legal rules*, Kluwer Academic Publishers, Dordrecht, 1998.
18. Royakkers, L.M.M. and F. Dignum, From collective to individual obligations, in: *Proceedings of The Law in the Information Society*, Florence, Instituto per la documentazione giuridica del CNR, pp. 1008-1022, 1998.
19. Santos, F., A.J.I. Jones and J. Carmo, Responsibility for action in organizations: a formal model, in: G. Hölstrom-Hintikka and R. Tuomela (eds.), *Contemporary Action Theory*, Vol II (Social Action), Synthese Library, Kluwer Academic Publishers, Dordrecht, pp. 333-350, 1997.
20. Singh, M.P., Multiagent Systems: A theoretical Framework for intentions, know-how, and communication, *Lecture Notes in Computer Science*, vol. 799, Berlin, Springer, 1995.
21. Vogel Carey, T., How to confuse commitment with obligation, *The Journal of Philosophy*, pp. 276-284, 1975.

Information Retrieval with Conceptual Graph Matching

Manuel Montes-y-Gómez [1], Aurelio López-López [2], Alexander Gelbukh [1]

[1] Center for Computing Research (CIC), National Polytechnic Institute (IPN),
Av. Juan Dios Bátiz s/n esq. Mendicabal, col. Zacatenco, CP. 07738, DF, Mexico.
mmontesg@susu.inaoep.mx, gelbukh@cic.ipn.mx

[2] INAOE. Luis Enrique Erro No. 1, Tonantzintla, Puebla, 72840 México.
allopez@inaoep.mx

Abstract. The use of conceptual graphs for the representation of text contents in information retrieval is discussed. A method for measuring the similarity between two texts represented as conceptual graphs is presented. The method is based on well-known strategies of text comparison, such as Dice coefficient, with new elements introduced due to the bipartite nature of the conceptual graphs. Examples of the representation and comparison of the phrases are given. The structure of an information retrieval system using two-level document representation, traditional keywords and conceptual graphs, is presented.

1. Introduction*

In many application areas of text analysis, for instance, in information retrieval and in text mining, shallow representations of texts have been recently widely used. In information retrieval, such shallow representations allow for a fast analysis of the information and a quick respond to the queries. In text mining, such representations are used because they are easily extracted from texts and easily analyzed.

Recently in all text-oriented applications, there is a tendency to begin using more complete representations of texts than just keywords, i.e., the representations with more types of textual elements. For instance, in information retrieval, these new representations increase the precision of the results; in text mining, they extend the kinds of discovered knowledge.

A method for the comparison of texts in such a representation is one of the main prerequisites to begin using the new representation in various applications of text processing. In this paper, we discuss the use of the conceptual graphs to represent the contents of documents for information retrieval and text mining. This representation incorporates the information about both the concepts mentioned in the text and their relationships, e.g., *[binary]* ← *(attr)* ← *[search]*. We present a method for measuring the similarity between two phrases represented as conceptual graphs. This method does not depend on the kind of concepts and relations used in the graphs.

First, we discuss the previous works concerning the comparison between two texts, introduce the notion of the conceptual graph, and describe the process of transformation of a text to a set of conceptual graphs. Then, we explain our method of compari-

* The work was done under partial support of CONACyT, REDII, and SNI, Mexico.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 312–321, 2000.

son of two conceptual graphs, with the corresponding formulae. Finally, we discuss possible applications of the method for information retrieval, and give an exa mple.

2. Related work

The comparison of text representations has been widely discussed in the literature. Important related work has been done in information retrieval, document clustering, conceptual clustering, and recently in text mining.

In information retrieval and document clustering, the weighted-keyword representation of documents is one of the most widely used [8, 9]. For this type of document representation, many different similarity measures are proposed, for instance, the Dice coefficient, the Jaccard coefficient, the Cosine coefficient [8], etc.

For the representation with binary term weights, the Dice coefficient is calculated as follows:

Dice coefficient: $$S_{D_1,D_2} = \frac{2n(D_1 \cap D_2)}{n(D_1)+n(D_2)}$$

where $n(D_i)$ is the number of terms in D_i, and $n(D_i \cap D_j)$ is the number of terms that the two documents D_i and D_j have in common.

Because of its simplicity and normalization, we take it as the basis for the similarity measure we propose.

In information retrieval, some other kinds of representations different from the keyword representation have been used, for instance, conceptual graphs [2, 7]. For these representations, different similarity measures have been described for comparing the query graph and the document graphs. One of the main comparison criteria used for conceptual graphs is that if a query graph is completely contained in the document graph, then the given document is relevant for the given query. This criterion means that the contents of a document must be more particular than the query, for the document to be relevant for the query.

In text mining, text representations and similarity measures borrowed from information retrieval are widely used. Other representations and measures, for instance, probability distributions of topics and other statistical parameters, have been used too [1, 5]. Usually such representations and measures are tuned to improve the discovering of knowledge in texts.

3. Conceptual graphs

To compare two texts, e.g., a document and the user's query, first their representations in the form of conceptual graphs are built. A conceptual graph is a network of concept nodes and relation nodes [10, 11]. The concept nodes represent entities, attributes, or events (actions); they are denoted with brackets. The relation nodes ide ntify the kind of relationship between two concept nodes; they are denoted with parentheses. At present, we consider relations from a few basic types, such as *attribute*,

subject, *object*, etc. Thus, a phrase *John loves Mary* is represented with a graph like [John] ← (subj) ← [love] → (obj) → [Mary], and not like [John] ← (love) → [Mary].

In the system we developed, to build a conceptual graph representation of a phrase, a part-of-speech tagger, a syntactic parser, and a semantic analyzer are used. For example, given the phrase

Algebraic formulation of flow diagrams,

first, the part-of-speech tagger supplies each word with a syntactic-role tag, given after the bar sign:[1]

Algebraic|JJ formulation|NN of|IN flow|NN diagrams|NNS .|.

Then a syntactic parser generates its structured representation:[2]

[[np, [n, [*formulation*, sg]], [adj, [*algebraic*]], [*of*,
[np, [n, [*diagram*, pl]], [n_pos, [np, [n, [*flow*, sg]]]]]], '.'].

The semantic analyzer generates one or more conceptual graphs out of such syntactic structure:[3]

[algebraic] ← (attr) ← [formulation] → (of) → [flow-diagram:*]

In this graph, the concept nodes represent the elements mentioned in the text, for example, nouns, verbs, adjectives, and adverbs, while the relation nodes represent some kind of relation between the concepts (prepositions are maintained to avoid the difficult problem of resolving the semantic relation they express). At the moment, we use only a limited set of relations but we plan to extend it to include some domain-specific semantic relations, and to start using more elements of the conceptual graph formalism, for instance, *n*-ary relations and contexts.

4. Comparison of conceptual graphs

For the purposes of information retrieval or text mining, it is important to be able to compare two phrases or texts represented with conceptual graphs. In particular, in information retrieval one of the text is the document and the other one is the user's query. Each phrase or text may be represented as a set of conceptual graphs, for instance, a long phrase, or a whole text consisting of many phrases, are represented as a set of conceptual graphs.

In general terms, our algorithm for the comparison of two conceptual graph representations of two texts consists of two main parts:

1. Find the intersection of the two (set of) graphs,
2. Measure the similarity between the two (set of) graphs as the relative size of each one of their intersection graphs.

---

[1] The tagger we use is based on the Penn Treebank tagset.

[2] The parser we use was developed by Tomek Strzalkowski of the New York University basing on The Linguist String Proyect (LSP) grammar designed by Naomi Sager.

[3] We do not discuss here the structure of the semantic analyzer we use.

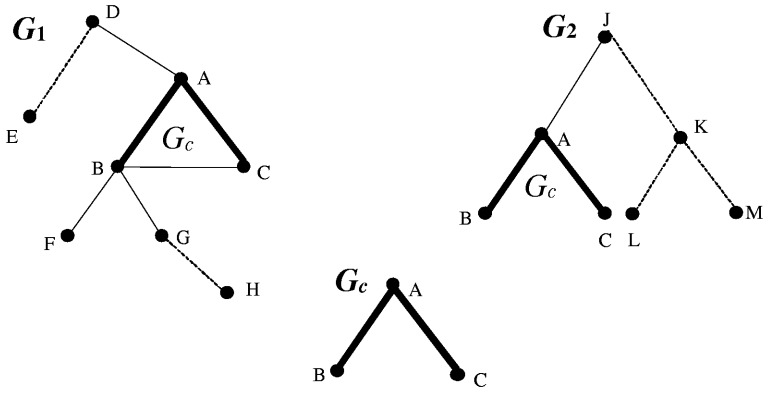


Fig.1. Intersection of two conceptual graphs

In general, we can find more than one subgraph as the intersection of the initial graphs, but the measurement of similarity is applied to each one of them separately, and only the highest value is kept. For the sake of explanation, we hereon deal with only one intersection subgraph.

In the first step, we build the intersection $G_1 \cap G_2 = G_c$ of the two original conceptual graphs G_1 and G_2. This intersection consists of the following elements:

- All concept nodes that appear in both original conceptual graphs G_1 and G_2;
- All relation nodes that appear in both G_1 and G_2 and relate the same concept nodes.

An example of such an intersection is shown on Figure 1. We show the concept nodes such as [*John*] or [*love*] as the points *A*, *B*, etc., and the relation nodes such as (subj) or (obj) as arcs. In the figure, of the concept nodes *A*, *B*, *C*, *D*, *E*, etc., only the concepts *A*, *B*, and C belong to both graphs G_1 and G_2. Though three arcs *A* — *B*, *A* — *C*, and *B* — *C* are present between these concepts in G_1, only two of them are present in both graphs (with bold lines). Of course, for the arc between two common concepts to be included in the G_c, it should have the same label and orientation (not shown in Figure 1) in the two original graphs.

In the second step, we measure the similarity between the graphs G_1 and G_2 based on their intersection graph G_c. The similarity measure is a value between 0 and 1, where 0 indicates that there is no similarity between the two texts, and 1 indicates that the two texts are semantically equivalent.

Because of the bipartite (concepts and relations) nature of the conceptual graph representations, the similarity measure is defined as a combination of two types of similarity: the conceptual similarity and the relational similarity:

- The conceptual similarity measures how similar the concepts and actions mentioned in both texts are (like topical comparison).
- The relational similarity measures the degree of similarity of the information about these concepts (concept interrelations) communicated in the two texts.

5. Similarity measure

Given two texts represented by the conceptual graphs G_1 and G_2 respectively and one of their intersection graphs G_c, we define the similarity s between them as a combination of two values: their conceptual similarity s_c and their relational similarity s_r.

The conceptual similarity s_c expresses how many concepts the two graphs G_1 and G_2 have in common. We calculate it using an expression analogous to the well-known Dice coefficient [8]:

$$s_c = \frac{2n(G_c)}{n(G_1)+n(G_2)}$$

where $n(G)$ is the number of concept nodes of a graph G. This expression varies from 0 when the two graphs have no concepts in common to 1 when the two graphs consist of the same set of concepts.

The relational similarity s_r indicates how similar the relations between the same concepts in both graphs are, that is, how similar the information communicated in both texts about these concepts is. In a way, it shows how similar the contexts of the common concepts in both graphs are.

We define the relational similarity s_r to measure the proportion between the degree of connection of the concept nodes in G_c, on the one hand, and the degree of connection of the same concept nodes in the original graphs G_1 and G_2, on the other hand. With this idea, a relation between two concept nodes conveys less information about the context of these concepts if they are highly connected in the original graphs, and conveys more information when they are weakly connected in the original graphs. We formalize this using a modified formula for the Dice coefficient:

$$s_r = \frac{2m(G_c)}{m_{G_c}(G_1)+m_{G_c}(G_2)}$$

where $m(G_c)$ is the number of the arcs (the relation nodes in the case of conceptual graphs) in the graph G_c, and $m_{G_c}(G_i)$ is the number of the arcs in the immediate neighborhood of the graph G_c in the graph G_i. The immediate neighborhood of $G_c \subseteq G_i$ in G_i consists of the arcs of G_i with at least one end belonging to G_c.

Figure 2 illustrates these measures. In this figure, the nodes *A*, *B* and *C* are the conceptual nodes common for G_1 and G_2 and thus belonging to G_c. Bold lines represent the arcs (relation nodes) common to the two graphs. The arcs marked with the symbol ✓ constitute the immediate neighborhood of the graph G_c (highlighted areas), their number is expressed by the term $m_{G_c}(G_i)$ in the formula above.

The value of $m_H(G)$ for a subgraph $H \subseteq G$ in practice can be calculated as: $m_H(G) = \sum_{c \in H} deg_G c - m(H)$, where $deg_G c$ is the degree of concept node c in the graph G, i.e., the number of the relation nodes connected to the concept node c in the graph G, and $m(H)$ is the number of relation nodes in the graph H.

Now that we have defined the two components of the similarity measure, s_c and s_r, we will combine them into a cumulative measure s. First, the combination is to be roughly multiplicative, for the cumulative measure to be roughly proportional to each

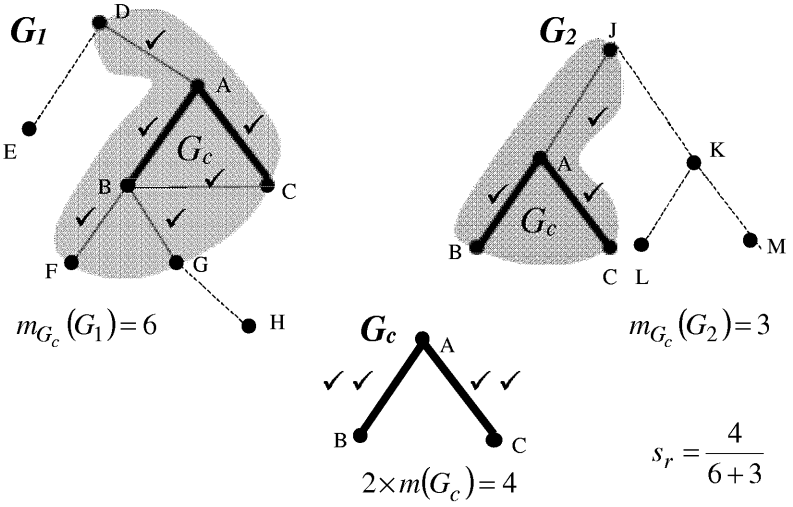


Fig. 2. Calculation of relational similarity

of the two components. This would give the formula $s=s_c\times s_r$. However, we can note that the relational similarity has a secondary importance, because it existence depends of the existence of some common concepts nodes, and because even if no common relations exist between the common concepts of the two graphs, some level of similarity exists between the two texts. Thus, while the cumulative similarity measure is proportional to s_c, it still should not be zero when $s_r = 0$. So we will smooth the effect of s_r:

$$s=s_c\times(a+b\times s_r).$$

With this definition, if no relational similarity exists between the graphs, that is, when $s_r=0$, the general similarity only depends of the value of the conceptual similarity. In this situation, the general similarity is a fraction of the conceptual similarity, where the coefficient a indicates the value of this fraction.

The values of the coefficients a and b depend on the structure of the graphs G_1 and G_2 (i.e. their value depend on the degree of connection of the elements of G_c in the original graphs G_1 and G_2). We calculate the values of a and b as follows:

$$a=\frac{2n(G_c)}{2n(G_c)+m_{G_c}(G_1)+mG_c(G_2)},$$

where $n(G_c)$ is the number of concept nodes in G_c and $m_{G_c}(G_1)+m_{G_c}(G_2)$ is the number of relation nodes in G_1 and G_2 that are connected to the concept nodes appearing in G_c.

With this formula, when $s_r=0$, then $s=a\times s_c$, that is, the general similarity is a fraction of the conceptual similarity, where the coefficient a indicates this portion.

Thus, the coefficient a expresses the percentage of information contained only in the concept nodes (according to their surrounding). It is calculated as the proportion between the number of common concept nodes (i.e. the concept nodes of G_c) and the total number of the elements in the context of G_c (i.e., all concept nodes of G_c and all relation nodes in G_1 and G_2 connected to the concept nodes that belong to G_c).

When $s_r = 1$, all information around the common concepts is identical and therefore they express the same things in both texts. In this situation, the general similarity takes it maximal similarity value $s = s_c$, and consequently $a + b \times s_r = 1$. Thus, the coefficient b is equal to $1 - a$.

6. Uses in information retrieval

Nowadays, with the electronic information explosion caused by Internet, increasingly diverse information is available. To handle and use such great amount of information, improved search engines are necessary. The more information about documents is preserved in their formal representation used for information retrieval, the better the documents can be evaluated and eventually retrieved.

Based on these ideas, we are developing a new information retrieval system. This system performs the document selection taking into account two different levels of document representation.

The first level is the traditional keyword document representation. It serves to select all documents potentially related to the topic(s) mentioned in the user's query. The second level is formed with the conceptual graphs reflecting some document details, for instance, the document intention. This second level complements the topical information about the documents and provides a new way to evaluate the relevance of the document for the query.

Figure 3 shows the general architecture of our information retrieval system with two-level document selection. In this system, the query-processing module analyses the query and extracts from it a list of topics (keywords). The keyword search finds all relevant documents for such a keyword-only query. Then, the information extraction module constructs the conceptual graphs of the query and the retrieved documents, according to the process described in section 3. This information is currently extracted from titles [6] and abstracts [4] of the documents. These conceptual graphs describe mainly the intention of the document, but they can express other type of relations, such as cause-effect relations [3].

The following example is a conceptual graph extracted from a document abstract.

[demonstrate] → (obj) → [validity: #] → (of) → [technique: #]

This graph indicates that the document in question has the intention of *demonstrating the validity of the technique.*

Then the query conceptual graph is compared – using the method described in this paper – with the graphs for the potentially relevant documents. The documents are then ordered by their value s of the similarity to the query.

After this process the documents retrieved at the beginning of the list will not only mention the key-topics expressed in the query, but also describe the intentions specified by the user.

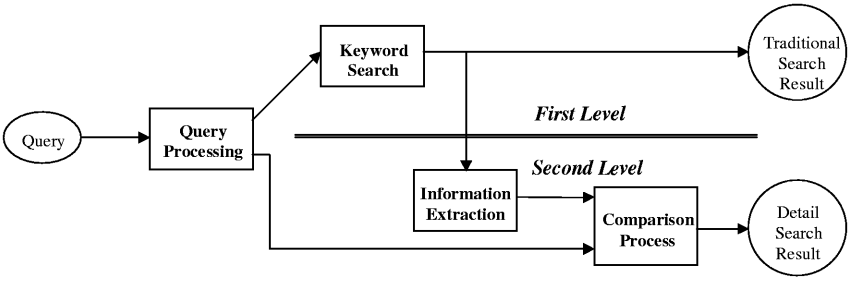


Fig. 3. Calculation of relational similarity.

This technique allows improving information retrieval in two main directions:

1. It permits to search the information using not only topical information, but also extratopical, for instance, the document intentions.
2. It produces a better raking of those documents closer to the user needs, not only in terms of subject.

7. Preliminary experimental results

In our experiments on information retrieval, we used a toy database of 512 documents randomly selected from one of the standard test document collections, namely CACM-3204. For each document, the conceptual graph of its title was constructed as described in Section 3. The graphs we used represented mainly the syntactic level of the texts. The relations used in the graphs were the following: *obj* (verb → its object), *subj* (verb → its subject), *attr* (noun or verb → its attribute: adjective or adverb) and prepositions (each specific prepositions, such as *of*).

For each query, we constructed its conceptual graph and compared it with the conceptual graphs of the document titles using the comparison method described above. The documents that had the highest similarity values with the query were selected as the search results.

For example, for the query "*description of a fast procedure for solving a system of linear equations*", we built the following conceptual graph:

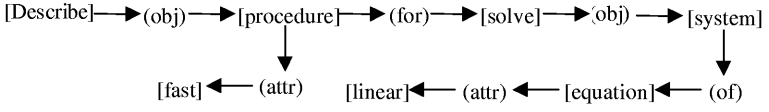


Table 1 illustrates the comparison with the following documents found:

(1) The document 2642:

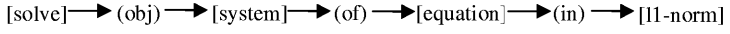
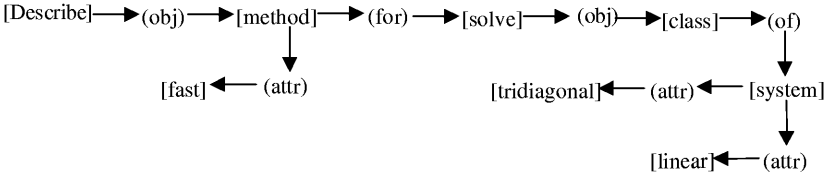


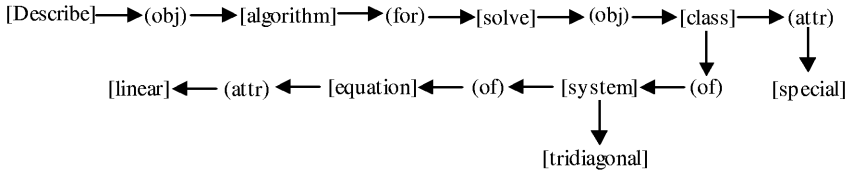
Table 1. An IR experiment using CG.

Graph	G_q	s_c	α	s_r	S
2642	[solve]→(obj)→[system]→(of)→[equation]	0.5	0.42	**0.5**	**0.357**
2697	[describe] [fast] [solve] [system] [linear]	**0.53**	0.42	0	0.224
1910	[describe] [solve] [system]→(of)→[equation]→(attr)→[linear]	0.5	**0.44**	**0.5**	0.333

(2) The document 2697:



(3) The document 1910:



This example shows the main properties of the measure and how the conceptual and relational similarities are combined to produce the final measure. One can see that our measure really scores the graphs with connected common elements higher than the ones with even a larger number of the nodes in common if they are not connected. Thus, our similarity measure focuses on what the text tells about the concepts (interconnection of concepts) rather than only on the concepts it mentions per se.

8. Conclusions

We have described the structure of an information retrieval system that uses the comparison of the document and the query represented with conceptual graphs to improve the precision of the retrieval process by better ranking on the results. In particular, we have described a method for measuring the similarity between conceptual graph representations of two texts. This method incorporates some well-known characteristics, for instance, the idea of the Dice coefficient – a widely used measure of similarity for the keyword representations of texts. It also incorporates some new characteristics derived from the conceptual graph structure, for instance, the combination of two complementary sources of similarity: the conceptual similarity and the relational similarity.

This measure is appropriate for text comparison because it considers not only the topical aspects of the phrases (difficult to obtain from short texts) but also the relationships between the elements mentioned in the texts. This approach is especially good for short texts. Since in information retrieval, in any comparison operation at least one of the two elements, namely, the query, is short, our method is relevant for information retrieval.

Currently, we are adapting this measure to use a concept hierarchy given by the user, i.e. an *is-a* hierarchy, and to consider some language phenomena as, for example, synonymy.

However, the use of the method of comparison of the texts using their conceptual graph representations is not limited by information retrieval. Other uses of the method include text mining and document classification.

References

1. Feldman, R., and I. Dagan (1995). "Knowledge Discovery in Textual databases (KDT)". 1st International conference on Knowledge discovery (KDD_95), pp.112-117, Montreal, 1995.
2. Genest D., and M. Chein (1997). "An Experiment in Document Retrieval Using Conceptual Graphs". Conceptual structures: Fulfilling Peirce's Dream. LNAI 1257, Springer, 1997.
3. Khoo, Christopher Soo-Guan (1997). "The Use of Relation Matching in Information Retrieval". Electronic Journal ISSN 1058-6768, September 1997.
4. López-López, Aurelio, and Sung H. Myaeng (1996). "Extending the capabilities of retrieval systems by a two level representation of content". Proceedings of the 1st Australian Document Computing Symposium, 1996.
5. Montes-y-Gómez, M., A. López-López, A. Gelbukh (1999a). "Text Mining as a Social Thermometer". In Procs. Workshop on Text Mining: Foundations, Techniques and Applications, Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden, August 1999.
6. Montes-y-Gómez, M., A. Gelbukh, A. López-López (1999b). "Document Title Patterns in Information Retrieval", Proc. of the Workshop on Text, Speech and Dialogue TDS'99, Plzen, Czech Republic, September 1999.
7. Myaeng, Sung H. (1990). "Conceptual Graph Matching as a Plausible Inference Technique for Text Retrieval". Proc. of the 5th Conceptual Structures Workshop, held in conjunction with AAAI-90, Boston, Ma, 1990.
8. Rasmussen, Edie (1992). "Clustering Algorithms". Information Retrieval: Data Structures & Algorithms. William B. Frakes and Ricardo Baeza-Yates (Eds.), Prentice Hall, 1992.
9. Salton, G. (1983). "Introduction to Modern Information Retrieval". McGraw Hill, 1983.
10. Sowa, John F. (1983). "Conceptual Structures: Information Processing in Mind and Machine". Ed. Addison-Wesley, 1983
11. Sowa, John F. (1999). "Knowledge Representation: Logical, Philosophical and Computational Foundations". 1st edition, Thomson Learning, 1999.

Approximate Pattern Matching in Shared-Forest

M. Vilares, F.J. Ribadas, and V.M. Darriba

Computer Science Department, Campus de Elviña s/n, 15071 A Coruña, Spain
vilares@udc.es ribadas@mail2.udc.es darriba@mail2.udc.es
WWW home page: http://www.dc.fi.udc.es/~vilares

Abstract. We present a proposal intended to demonstrate the applicability of tabulation techniques to pattern recognition problems, when dealing with structures sharing some common parts. This work is motivated by the study of information retrieval for textual databases, using pattern matching as a basis for querying data.

1 Introduction

One critical aspect of an information system that determines its effectiveness is indexing, that is, the representation of concepts to get a well formed data structure for search. Once this has been done, the system must define a strategy in order to translate a query statement to the database and determine the information to be returned to the user.

Until recently, indexing was accomplished by creating a bibliographic citation in a structured file that references the original text. This approach allows a reduction in time and space bounds, although the ability to find information on a particular subject is limited by the system which creates index terms for that subject. At present, the significant reduction in cost processing has propitiated the notion of total document indexing, for which all words in a document are potential index descriptors. This reduction saves the indexer from entering index terms that are identical to words in the document, but does not facilitate the finding of relevant information for the user.

In effect, the words used in a query do not always reflect the value of the concepts being presented. It is the combination of these words and their semantic implications that contain the value of these concepts which leads us to more sophisticated index representations, such as context-free grammars [1] [2]. This information is inherent in the document and query. So, matching becomes a possible mechanism for extracting a common pattern from multiple data and we could use it to locate information of linguistic interest in natural language processing. Some related work about using syntax structures in IR can be found in Smeaton et al. [3] [4] [5].

However, the language intended to represent the document can often only be approximately defined, and therefore ambiguity arises. Since it is desirable to consider all possible parses for semantic processing, it is convenient to merge parse trees as much as possible into a single structure that allows them to share common parts. Although in the case of the query, language ambiguity could

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 322–333, 2000.

probably be eliminated, queries could vary widely from indexes and an approximate matching strategy becomes necessary. At this point, our aim is to exploit structural sharing during the matching process in order to improve performances.

2 The Editing Distance

Given trees, T_1 and T_2, we define an *edit operation* as a pair $a \rightarrow b$, $a \in \text{labels}(T_1) \cup \{\varepsilon\}$, $b \in \text{labels}(T_2) \cup \{\varepsilon\}$, $(a,b) \neq (\varepsilon,\varepsilon)$, where ε represents the empty string. We can delete a node $(a \rightarrow \varepsilon)$, insert a node $(\varepsilon \rightarrow b)$, and change a node $(a \rightarrow b)$. Each edit operation has an associated cost, $\gamma(a \rightarrow b)$, that we extend to a sequence S of edit operations $s_1, s_2, \ldots, s_n$ in the form $\gamma(S) = \sum_{i=1}^{|S|}(\gamma(s_i))$. The distance between T_1 and T_2 is defined by the metric:

$$\delta(T_1, T_2) = \min\{\gamma(S),\ S \text{ editing sequence taking } T_1 \textit{ to } T_2\}$$

Given a postorder traversal, as shown in Fig. 2, to name each node i of a tree T by $T[i]$, a *mapping* from T_1 to T_2 is a triple (M, T_1, T_2), where M is a set of integer pairs (i,j) satisfying, for each $1 \leq i_1, i_2 \leq |\ T_1\ |$ and $1 \leq j_1, j_2 \leq |\ T_2\ |$:

$i_1 = i_2$	**iff** $j_1 = j_2$
$T_1[i_1]$ is to the left of $T_1[i_2]$	**iff** $T_2[j_1]$ is to the left of $T_2[j_2]$
$T_1[i_1]$ is an ancestor of $T_1[i_2]$	**iff** $T_2[j_1]$ is an ancestor of $T_2[j_2]$

which corresponds, in each case, to one-to-one assignation, sibling order preservation and ancestor order preservation. The cost, $\gamma(M)$, of a mapping (M, T_1, T_2) is computed from relabeling, deleting and inserting operations, as follows:

$$\gamma(M) = \sum_{(i,j)\in M} \gamma(T_1[i] \rightarrow T_2[j]) + \sum_{i\in\mathcal{D}} \gamma(T_1[i] \rightarrow \varepsilon) + \sum_{j\in\mathcal{I}} \gamma(\varepsilon \rightarrow T_2[j])$$

where $\mathcal{D}$ and $\mathcal{I}$ are, respectively, the nodes in T_1 and T_2 not touched by any line in M. Tai [6] proves, given trees T_1 and T_2, that

$$\delta(T_1, T_2) = min\{\gamma(M),\ M \text{ mapping from } T_1 \text{ to } T_2\}$$

which allows us to focus on edit sequences that are a mapping. We show, in the leftmost diagram of Fig. 1, an example of mapping between two trees. The rightmost diagram includes a sequence of edit operations not constituting a mapping.

3 The Zhang and Shasha's Algorithm

We have based our work in the Zhang and Shasha's tree pattern matching algorithm, introduced in [8] and extended with advanced matching features in [9]. A major characteristic of this algorithm is its bottom-up oriented approach. Given the *l_keyroots*(T), the set of all nodes in T which have a left sibling plus the root,

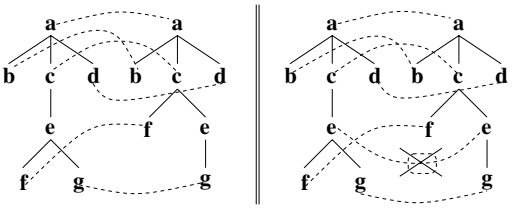


Fig. 1. An example on mappings

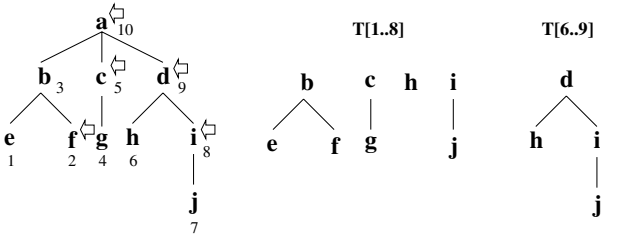


Fig. 2. The forest distance using a postorder numbering

root(T), of T; the algorithm proceeds through the nodes determining mappings from all leaf l_keyroots first, then all l_keyroots at the next higher level, and so on to the root.

We introduce $l(i)$ (resp. $anc(i)$) as the leftmost leaf descendant of the subtree rooted at T_i (resp. the ancestors of T_i) in a tree T, and $T[i..j]$ as the ordered sub-forest of T induced by the nodes numbered i to j inclusive. In particular, we have $T[l(i)..i]$ which is the tree rooted at $T[i]$, as shown in Fig. 2, where the set of *l_keyroots* is indicated by arrows. We also define the *forest edition distance* as a generalization of δ, in the form

$$\text{f\_d}(T_1[i_1..i_2], T_2[j_1..j_2]) = \delta(T_1[i_1..i_2], T_2[j_1..j_2])$$

that we shall denote *f_d*$(i_1..i_2, j_1..j_2)$ when the context is clear. Intuitively, this new concept computes the distance between two nodes, $T_1[i_2]$ and $T_2[j_2]$, in the context of their left siblings in the corresponding trees, while the corresponding tree distance, $\delta(T_1[i_2], T_2[j_2])$, is computed only from their descendants.

Formally, we compute *t_d*(T_1, T_2) applying the formulas that follow, for nodes $i_1 \in anc(i)$ and $j_1 \in anc(j)$, as illustrated in Fig. 3, taking into account the different cases:

$$\text{f\_d}(l(i_1)..i, l(j_1)..j) = \begin{cases} \min \left\{ \begin{array}{ll} \text{f\_d}(l(i_1)..i-1,\ l(j_1)..j) & +\gamma(T_1[i] \to \varepsilon), \\ \text{f\_d}(l(i_1)..i,\ l(j_1)..j-1) & +\gamma(\varepsilon \to T_2[j]), \\ \text{f\_d}(l(i_1)..i-1,\ l(j_1)..j-1) & +\gamma(T_1[i] \to T_2[j]) \end{array} \right\} \\ \textbf{iff } l(i) = l(i_1) \textbf{ and } l(j) = l(j_1) \\ \min \left\{ \begin{array}{ll} \text{f\_d}(l(i_1)..i-1,\ l(j_1)..j) & +\gamma(T_1[i] \to \varepsilon), \\ \text{f\_d}(l(i_1)..i,\ l(j_1)..j-1) & +\gamma(\varepsilon \to T_2[j]), \\ \text{f\_d}(l(i_1)..l(i)-1,\ l(j_1)..l(j)-1) & +\text{t\_d}(i,j) \end{array} \right\} \\ \textbf{otherwise} \end{cases}$$

Now, to compute the distance between T_1 and T_2, it will be sufficient to take into account that

$$\text{t\_d}(T_1, T_2) = \text{f\_d}(l(\text{root}(T_1))..\text{root}(T_1), l(\text{root}(T_2))..\text{root}(T_2))$$

The time complexity of this algorithm is, in the worst case:

$$\mathcal{O}(\mid T_1 \mid\mid T_2 \mid \min(\text{depth}(T_1), \text{leaves}(T_1))\min(\text{depth}(T_1), \text{leaves}(T_2)))$$

where $\mid T_1 \mid$ (resp. $\mid T_2 \mid$) is the number of nodes in the pattern tree T_1 (resp. in the data tree T_2), leaves(T_1) (resp. leaves(T_2)) is the number of leaves in T_1 (resp. in T_2), and depth(T_1) (resp. depth(T_2)) is the depth of T_1 (resp. of T_2).

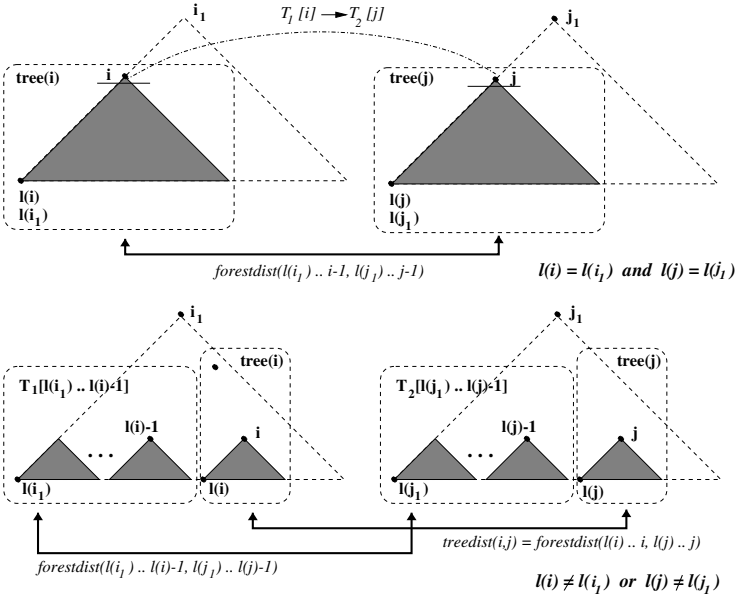


Fig. 3. The forest distance in Zhang and Shasha's algorithm

4 Relating Parsing and Approximate Tree Matching

The major question of Zhang and Shasha's algorithm [8], is the tree distance algorithm itself. However, parsing and tree-to-tree correction are topologically related and it is necessary to understand the mechanisms that cause the phenomenon of tree duplication to get the best performance.

A major factor to take into account is the syntactic representation used. We chose to work in the parsing context described for ICE [7]. Here, authors represent a parse as the chain of the context-free rules used in a leftmost reduction of the input sentence, rather than as a tree. When the sentence has distinct parses, the set of all possible parse chains is represented in finite shared form by a context-free grammar that generates that possibly infinite set.

This difference with most other parsers is only apparent, since context-free grammars can be represented by AND-OR graphs that in our case are precisely the shared-forest graph. In this graph, AND-nodes correspond to the usual parse-tree nodes, while OR-nodes correspond to ambiguities. Sharing of structures is represented by nodes accessed by more than one other node, and it may correspond to sharing of a complete subtree, but also to sharing of a part of the descendants of a given node.

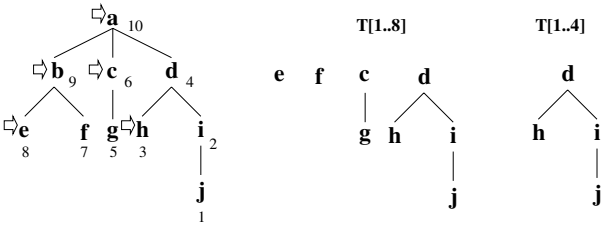


Fig. 4. The forest distance using an inverse postorder numbering

In this context, sharing of a tail of sons in a node of the resulting forest is possible. More exactly, bottom-up parsing may share only the rightmost constituents, while top-down parsing may only share the leftmost ones. This relates to the type of search used to build the forest. Breadth first search results in bottom-up constructions and depth first search results in top-down ones, as is shown in Fig. 5.

At this level, one major observation we noted is that Zhang and Shasha consider a postorder traversal, computing the forest distance by left-recursion on this search. As a consequence, we would need to consider a top-down parsing architecture to avoid redundant computations. However, top-down parsers are not computationally efficient, and a bottom-up approach, as is the case of ICE, requires a rightmost search of tree constituents. This implies redefining the architecture of the original matching strategy.

To accomplish this change, nodes in a tree T will be first numbered considering an inverse postorder traversal, as is shown in Fig. 4. We also introduce

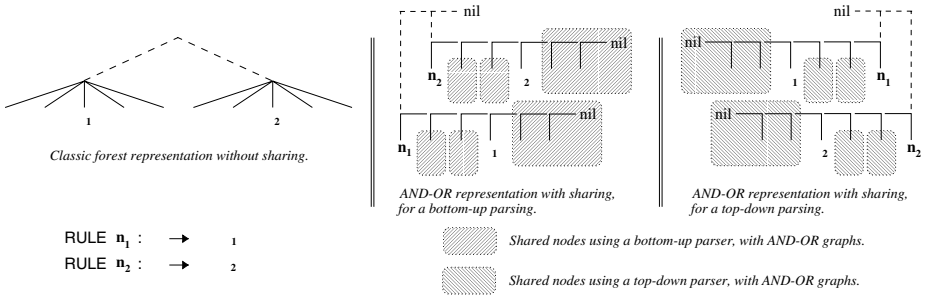


Fig. 5. How shared forest are built using an AND-OR formalism

r_keyroots(T), indicated by arrows in Fig. 4, as the set of all nodes in a tree T which have a right sibling plus the root, *root*(T), of T. And also we define $r(i)$ as the rightmost leaf descendant of the subtree rooted at T_i in a tree T.

From here, the alternative construction for the forest edition distance is analogous to the original algorithm, as shown in Fig. 6. For a better understanding we shall present the computations used with the inverse postorder. Given trees T_1 and T_2, and nodes $i_1 \in anc(i)$ and $j_1 \in anc(j)$, we have then that:

$$\text{f\_d}(r(i_1)..i, r(j_1)..j) = \begin{cases} \min \left\{ \begin{array}{ll} \text{f\_d}(r(i_1)..i-1,\ r(j_1)..j) & +\gamma(T_1[i] \to \varepsilon), \\ \text{f\_d}(r(i_1)..i,\ r(j_1)..j-1) & +\gamma(\varepsilon \to T_2[j]), \\ \text{f\_d}(r(i_1)..i-1,\ r(j_1)..j-1) & +\gamma(T_1[i] \to T_2[j]) \end{array} \right\} \\ \textbf{iff } r(i) = r(i_1) \textbf{ and } r(j) = r(j_1) \\ \\ \min \left\{ \begin{array}{ll} \text{f\_d}(r(i_1)..i-1,\ r(j_1)..j) & +\gamma(T_1[i] \to \varepsilon), \\ \text{f\_d}(r(i_1)..i,\ r(j_1)..j-1) & +\gamma(\varepsilon \to T_2[j]), \\ \text{f\_d}(r(i_1)..r(i)-1,\ r(j_1)..r(j)-1) & +\text{t\_d}(i,j) \end{array} \right\} \\ \textbf{otherwise} \end{cases}$$

To compute $t\_d(T_1, T_2)$ it will be sufficient to take into account that

$$\text{t\_d}(T_1, T_2) = \text{f\_d}(\text{root}(T_1)..r(\text{root}(T_1)), \text{root}(T_2)..r(\text{root}(T_2)))$$

Lastly, time and space bounds are the same as in the classic Zhang and Shasha's algorithm.

5 Approximate Matching in Shared Forest

We now offer a simple explanation of how both environments, parsing and approximate tree matching, can be efficiently integrated in practice.

To start with, let T_1 be a labeled ordered tree, and T_2 an AND-OR graph, both of them built using our parsing frame. We shall identify T_1 with a query and T_2 with a part of the syntactic representation for a textual database with a certain degree of ambiguity. The presence of OR nodes in T_2 has two main

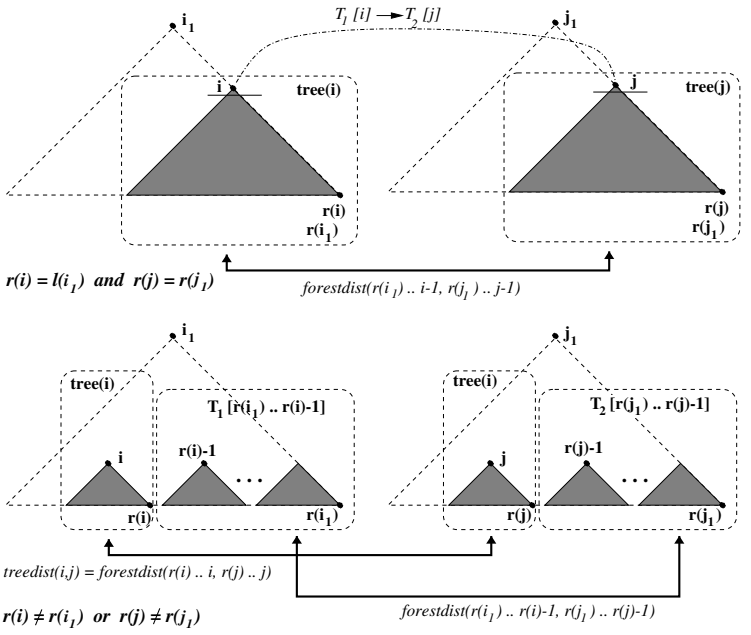


Fig. 6. The forest distance in our proposal

implications in our work: Firstly, there will exist situations where we must handle simultaneous values for some forest distances and, secondly, the parser may share some structures among the descendants of the different branches in an OR node. We shall now present the manner in which we calculate the distance between a pattern tree and the set of trees that are represented within the AND-OR graph, and how to take advantage of the shared structures created by the parser. The time complexity of this algorithm will be, in the worst case:

$$\mathcal{O}(\mid T_1 \mid\mid T_2 \mid \min(\text{depth}(T_1), \text{leaves}(T_1))\min(\text{depth}(T_2), \text{leaves}(T_2)))$$

where now $\mid T_2 \mid$ is the maximum number of nodes for a tree in T_2, $\text{leaves}(T_2)$ is the maximum number of leaves for a tree in T_2, and $\text{depth}(T_2)$ is the maximum depth for a tree in T_2.

Let $T_1[i]$ be the current node in the inverse postorder for T_1 and $i_1 \in \text{anc}(i)$ a r_keyroot. Given an OR node $T_2[k]$ we can distinguish two situations, depending on the situation of this OR node and the situation of the r_keyroots of T_2.

5.1 Sharing into a same r_keyroot

Let $T_2[j']$ and $T_2[j'']$ be the nodes we are dealing with in parallel for two branches labeled $T_2[k']$ and $T_2[k'']$ of the OR node $T_2[k]$. We have that $j_1 \in anc(j') \cap anc(j'')$, that is, the tree rooted at the r_keyroot $T_2[j_1]$ includes the OR alternatives $T_2[k']$ and $T_2[k'']$.

Such a situation is shown in Fig. 7 using a classic representation and the AND-OR graphs. Here, lightly shaded part refers to nodes whose distance have been computed in the inverse postorder before the OR node $T_2[k]$. The heavily shaded part represents a shared structure. The notation "• • •" in figures representing AND-OR graphs, expresses the fact that we descend along the rightmost branch of the corresponding tree.

We shall assume that nodes $T_2[r(j')-1]$ and $T_2[r(j'')-1]$ are the same, that is, their corresponding subtrees are shared. So, $T_2[r(j')]$ (resp. $T_2[r(j'')]$) is the following node in $T_2[k']$ (resp. $T_2[k'']$) to be dealt with once the distance for the shared structure has been computed.

At this point, our aim is to compute the value for $f\_d(r(i_1)..i,\ r(j_1)..\hat{\jmath})$, $\hat{\jmath} \in \{j',\ j''\}$, proving that we can translate parse sharing into sharing on computations for these distances.

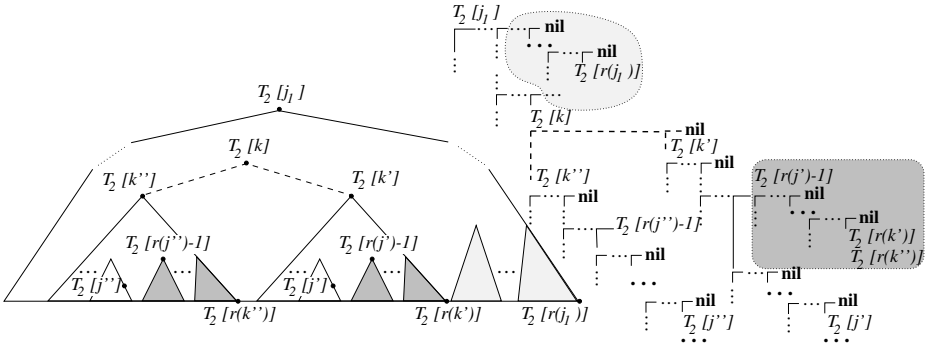


Fig. 7. Sharing into a same r_keyroot

Formally, the values for $f\_d(r(i_1)..i,\ r(j_1)..\hat{\jmath})$, $\hat{\jmath} \in \{j',\ j''\}$ are given by:

$$\mathrm{f\_d}(r(i_1)..i, r(j_1)..\hat{\jmath}) = \begin{cases} \min \left\{ \begin{array}{ll} \mathrm{f\_d}(r(i_1)..i-1,\ r(j_1)..\hat{\jmath}) & +\gamma(T_1[i] \to \varepsilon), \\ \mathrm{f\_d}(r(i_1)..i,\ r(j_1)..\hat{\jmath}-1) & +\gamma(\varepsilon \to T_2[\hat{\jmath}]), \\ \mathrm{f\_d}(r(i_1)..i-1,\ r(j_1)..\hat{\jmath}-1) & +\gamma(T_1[i] \to T_2[\hat{\jmath}]) \end{array} \right\} \\ \textbf{iff } r(i) = r(i_1) \textbf{ and } r(\hat{\jmath}) = r(j_1) \\ \min \left\{ \begin{array}{ll} \mathrm{f\_d}(r(i_1)..i-1,\ r(j_1)..\hat{\jmath}) & +\gamma(T_1[i] \to \varepsilon), \\ \mathrm{f\_d}(r(i_1)..i,\ r(j_1)..\hat{\jmath}-1) & +\gamma(\varepsilon \to T_2[\hat{\jmath}]), \\ \mathrm{f\_d}(r(i_1)..r(i)-1,\ r(j_1)..r(\hat{\jmath})-1) & +\mathrm{t\_d}(i,\hat{\jmath}) \end{array} \right\} \\ \textbf{otherwise} \end{cases}$$

where $\hat{\jmath} \in \{j',\ j''\}$. Here, $r(j_1) \neq r(\hat{\jmath})$, since we have assumed there is a shared structure between $T_2[r(\hat{\jmath})]$ and $T_2[r(j_1)]$. So, we can focus on the alternative computation, where:

1. The values for $f\_d(r(i_1)..i-1,\ r(j_1)..\hat{\jmath})$, $\hat{\jmath} \in \{j',\ j''\}$ have been computed by the approximate matching algorithm in a previous step. So, in this case, parse sharing has not consequences on the natural computation for the distances.

2. Two cases are possible in relation to the nature of nodes $T_2[\hat{\jmath}]$, $\hat{\jmath} \in \{j',\ j''\}$, these are:
   - If both nodes are leaves, then $r(\hat{\jmath}) = \hat{\jmath}$. As a consequence, we have that
   $$T_2[j'-1] = T_2[r(j')-1] = T_2[r(j'')-1] = T_2[j''-1]$$
   and the values $f\_d(r(i_1)..i,\ r(j_1)..\hat{\jmath}-1)$, $\hat{\jmath} \in \{j',\ j''\}$ are also the same.
   - Otherwise, following the inverse postorder, we would arrive at the rightmost leaves of $T_2[j']$ and $T_2[j'']$, where we could apply the reasoning considered in the previous case.
3. Values for the distances $f\_d(r(i_1)..r(i)-1,\ r(j_1)..r(\hat{\jmath})-1)$, $\hat{\jmath} \in \{j',\ j''\}$ are identical, given that nodes $T_2[r(\hat{\jmath})-1]$, $\hat{\jmath} \in \{j',\ j''\}$ are shared by the parser.

5.2 Sharing between different r_keyroots

We have that $j'_1 \in anc(j')$ and $j''_1 \in anc(j'')$, with $j'_1 \neq j''_1$, are two r_keyroots. We also have an OR node $T_2[k]$ being a common ancestor of these two nodes. We suppose that the r_keyroots are in different branches, that is, there exists a r_keyroot, $T_2[j'_1]$ (resp. $T_2[j''_1]$), in the branch labeled $T_2[k']$ (resp. $T_2[k'']$).

Our aim now is to compute the value for distances $f\_d(r(i_1)..i,\ r(\hat{\jmath}_1)..\hat{\jmath})$, where pairs $(\hat{\jmath}_1, \hat{\jmath})$ are in $\{(j'_1,\ j'),\ (j''_1, j''\ )\}$. Formally, we have that these values are given by:

$$\mathrm{f\_d}(r(i_1)..i, r(\hat{\jmath}_1)..\hat{\jmath}) = \begin{cases} \min \left\{ \begin{array}{ll} \mathrm{f\_d}(r(i_1)..i-1,\ r(\hat{\jmath}_1)..\hat{\jmath}) & +\gamma(T_1[i] \to \varepsilon), \\ \mathrm{f\_d}(r(i_1)..i,\ r(\hat{\jmath}_1)..\hat{\jmath}-1) & +\gamma(\varepsilon \to T_2[\hat{\jmath}]), \\ \mathrm{f\_d}(r(i_1)..i-1,\ r(\hat{\jmath}_1)..\hat{\jmath}-1) & +\gamma(T_1[i] \to T_2[\hat{\jmath}]) \end{array} \right\} \\ \textbf{iff } r(i) = r(i_1) \textbf{ and } r(\hat{\jmath}) = r(\hat{\jmath}_1) \\ \\ \min \left\{ \begin{array}{ll} \mathrm{f\_d}(r(i_1)..i-1,\ r(\hat{\jmath}_1)..\hat{\jmath}) & +\gamma(T_1[i] \to \varepsilon), \\ \mathrm{f\_d}(r(i_1)..i,\ r(\hat{\jmath}_1)..\hat{\jmath}-1) & +\gamma(\varepsilon \to T_2[\hat{\jmath}]), \\ \mathrm{f\_d}(r(i_1)..r(i)-1,\ r(\hat{\jmath}_1)..r(\hat{\jmath})-1) & +\mathrm{t\_d}(i,\hat{\jmath}) \end{array} \right\} \\ \textbf{otherwise} \end{cases}$$

The situation, shown in Fig. 8, makes possible $r(i) = r(i_1)$ and $r(\hat{\jmath}) = r(\hat{\jmath}_1)$. In this first case, we can assume that a tail of sons is shared by nodes $T[\hat{\jmath}]$, $\hat{\jmath} \in \{j',\ j''\}$. We can also assume that this tail is proper given that, otherwise, our parser guarantees that the nodes $T_2[\hat{\jmath}]$, $\hat{\jmath} \in \{j',\ j''\}$ are also shared.

Taking into account our parsing strategy, which identifies syntactic structures and computations, we conclude that the distances $f\_d(r(i_1)..i,\ r(\hat{\jmath}_1)..\hat{\jmath})$, with $(\hat{\jmath}_1, \hat{\jmath}) \in \{(j'_1,\ j'),; (j''_1, j''\ )\}$ do not depend on previous computations over the shared tail, such as is shown in the left-hand-side of Fig. 8. So, this sharing has no consequences on the calculus, although it will have effects on the computation of distances for nodes in the rightmost branch of the tree immediately to the left of the shared tail of sons, which is denoted by a double pointed line in Fig. 8, as we shall show immediately.

We consider now the second case, that is, the computation of the forest distance when $r(\hat{\jmath}_1) \neq r(\hat{\jmath})$, as is shown in Fig. 9. Here, in relation to each of the three alternative values to compute the minimum, we have that:

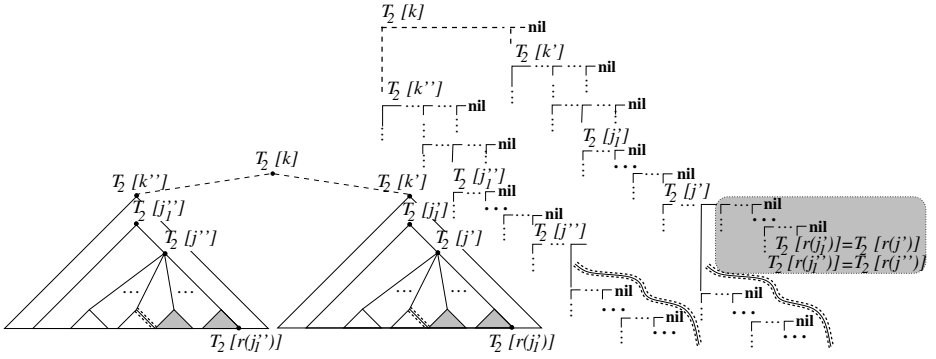


Fig. 8. Sharing between different r_keyroots (first case)

1. The values for $f\_d(r(i_1)..i-1,\ r(\hat{\jmath}_1)..\hat{\jmath}),\ (\hat{\jmath}_1,\hat{\jmath}) \in \{(j'_1,\ j'),\ (j''_1, j'')\}$ have been computed by the approximate matching algorithm in a previous step and parse sharing does not affect the computation for distances.
2. We distinguish two cases in relation to the nature of nodes $T_2[\hat{\jmath}],\ \hat{\jmath} \in \{(j',\ j'')$. We shall apply the same reasoning considered when we had an only r_keyroot:
   - If both nodes are leaves, then $r(\hat{\jmath}) = \hat{\jmath}$. As a consequence, we have that

$$T_2[j'-1] = T_2[r(j')-1] = T_2[r(j'')-1] = T_2[j''-1]$$

     and therefore the values for distances $f\_d(r(i_1)..i,\ r(\hat{\jmath}_1)..\hat{\jmath}-1)$ with $(\hat{\jmath}_1,\hat{\jmath}),\in \{(j'_1,\ j'),\ (j''_1, j'')\}$, are also the same.
   - Otherwise, following the inverse postorder, we arrive at the rightmost leaves of $T_2[j']$ and $T_2[j'']$, where we can apply the reasoning considered in the previous case.

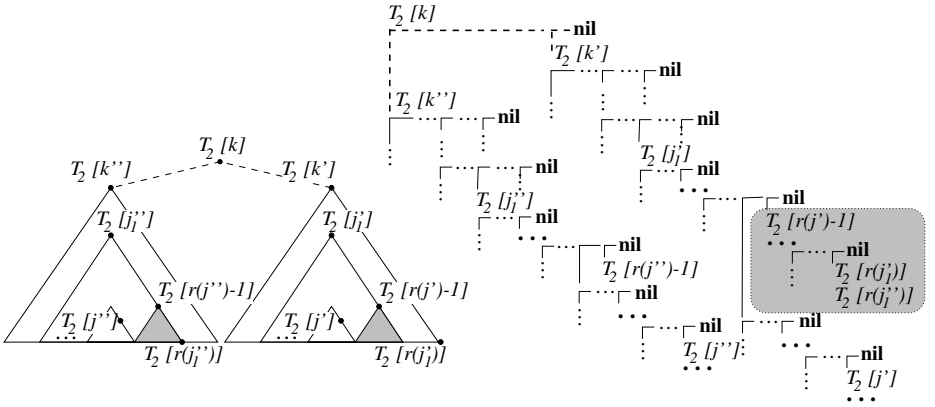


Fig. 9. Sharing between different r_keyroots (second case)

3. Values for the distances $f\_d(r(i_1)..r(i)-1,\ r(\hat{\jmath}_1)..r(\hat{\jmath})-1)$, $\hat{\jmath} \in \{j',\ j''\}$ are identical, given that the trees rooted by nodes $T_2[r(\hat{\jmath})-1]$, $\hat{\jmath} \in \{j',\ j''\}$ are shared by the parser.

6 Experimental Results

We consider the language of arithmetical expressions to illustrate the discussion, comparing our proposal with Tai [6], and Zhang and Shasha's algorithm [8]. We consider two deterministic grammars, $\mathcal{G}_L$ and $\mathcal{G}_R$, representing respectively the left and right associative versions for the arithmetic operators; and a non-deterministic one $\mathcal{G}_N$. We assume that parsers are built using ICE [7], and tests have been applied on data inputs of the form $a_1+a_2+\ldots+a_i+a_{i+1}$, with i even, representing the number of addition operators. In the non-deterministic case, these programs have a number of ambiguous parses which grows exponentially with i. This number is:

$$C_0 = C_1 = 1 \quad \text{and} \quad C_i = \binom{2i}{i} \frac{1}{i+1},\ \text{if } i > 1$$

As our pattern, we have used deterministic parse trees from inputs of the form $a_1 + b_1 + a_3 + b_3 + \ldots + b_{i-1} + a_{i-1} + b_{i+1} + a_{i+1}$, where $b_j \neq a_{j-1}$, for all $j \in \{1, 3, \ldots i-1, i+1\}$.

In the deterministic case, patterns are built from the left-associative (resp. right-associative) interpretation for $\mathcal{G}_L$ (resp. $\mathcal{G}_R$), which allows us to evaluate the impact of traversal orientation in the performance. So, Fig. 10 proves the adaptation of our proposal (resp. Zhang and Shasha's algorithm) to left-recursive (resp. right-recursive) derivations, which corroborates our conclusions. These tests also show the independence of Tai's algorithm from the grammar rules topology. This is due to the fact that mapping between two nodes is not computed from the mapping between their descendents, but from their ancestors, where structural sharing is not allowed by the parser. As a consequence, Tai's approach does not benefit from the dynamic programming architecture.

In the non-deterministic case, patterns are built from the left-associative interpretation of the query, which is not relevant given that rules in $\mathcal{G}_N$ are symmetrical. Here, we evaluate the gain in efficiency due to sharing of computations in a dynamic frame, as is also shown in Fig. 10.

7 Conclusions

Most classic approaches for dealing with tree-to-tree correction have been proposed in the context of dynamic programming algorithms. In turn, tabulation techniques are becoming a common way of dealing with highly redundant computations occurring, for instance, in natural language processing. However, no previous work has been developed in order to profit from this characteristic in the tree matching domain.

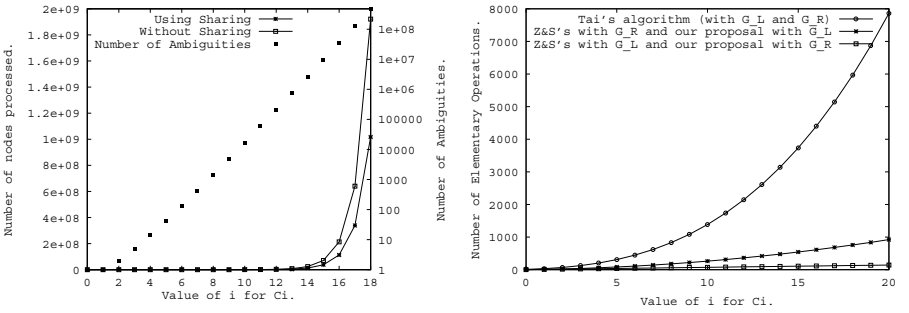


Fig. 10. Results on approximate tree matching

In this paper, we have proved that tree matching can be adapted to deal with shared forest in the context of information retrieval. To do this, the impact of the parsing strategy on the resulting shared structure should be studied. This allows formal justification of the type of traversal used to visit nodes during the matching, obtaining the maximum advantage from parse sharing.

References

1. Kilpelainen, P., Mannila, H.: Grammatical Tree Matching. Lecture Notes in Computer Science **644** (1992) 159–171.
2. Kilpelainen, P.: Tree Matching Problems with Applications to Structured Text Databases. Ph.D. Thesis, Department of Computer Science, University of Helsinki, 1992. Helsinki, Finland
3. Smeaton, Alan F. : Incorporating Syntactic Information into a Document Retrieval Strategy: An Investigation. Proc. the 9^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 103–113, 1986.
4. Smeaton, A.F. and van Rijsbergen, C.J.: Experiments on Incorporating Syntactic Processing of User Queries into a Document Retrieval Strategy. Proc. of the 11^{th} International ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 31-54. Grenoble, France, 1988.
5. Smeaton, A.F and O'Donell, R. anf Kelley,F.: Indexing Structures Derived from Syntax in TREC-3: System Description. Proc. of 3^{rd} Text REtrieval Conference (TREC-3), D.K. Harman (ed.), NIST Special Publication, 1994.
6. Tai, Kuo-Chung.: Syntactic error correction in programming languages. IEEE Transactions on Software Engineering **4(5)** (1978) 414–425.
7. Vilares, M., Dion, B.A.: Efficient incremental parsing for context-free languages. Proc. of the 5^{th} IEEE International Conference on Computer Languages (1994) 241–252, Toulouse, France.
8. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing (1989) **18** 1245–1262.
9. Zhang, K. and Shasha, D. and Wang, J.T.L. Approximate Tree Matching in the Presence of Variable Length Don't Cares. Journal of Algorithms, pages 33–66, vol **16 (1)**, 1994

A Data Model for Temporal XML Documents

Toshiyuki Amagasa Masatoshi Yoshikawa Shunsuke Uemura

Graduate School of Information Science
Nara Institute of Science and Technology
8916–5 Takayama, Ikoma 630–0101 Japan
{amagasa, yosikawa, uemura}@is.aist-nara.ac.jp

Abstract. XML is expected to become the next generation standard language for exchanging data over the Internet. In general, the contents of XML documents may change as time goes by, and then, it is important to capture entire histories of those documents. In this paper, we propose a logical data model for representing histories of XML documents. The proposed model extends the XPath data model, and is capable of representing change histories of XML documents. Various alternative approaches to the physical implementation of the model are also presented.

1 Introduction

XML [13] has attracted a great deal of public attention in recent years as a format for exchanging data over network as well as a meta language for exchanging structured documents. In addition, new technologies, such as WebDAV [12], are enabling users to collaboratively edit and manage structured documents on remote servers. In such applications, the contents of documents change over time. Hence, it is important to capture the entire histories of those documents.

Let us consider an example session of editing an XML document in Figure 1. This is an unfinished document as of January 1, 2000. Suppose the two authors, Taro and Jiro, edited the document as follows: (1) Jiro finished the second section in the first chapter on January 4; (2) Taro changed the title to "Introduction to XML" on January 10; (3) Jiro obtained his email address "jiro@db.aist-nara.ac.jp" on January 15; and (4) Taro removed the second section on January 20. After a series of these editorial process, we may want to restore the document at a particular previous state. In such a situation, recording histories of XML documents is quite useful.

In fact, we can record the changes of documents and restore the past states of them by using popular tools, such as *diff*, *RCS*, *SCCS*, and *CVS*. However, these tools are not adequate for XML documents, because XML permits some notations that represents the same content. In such a situation, *diff*-based tools described above are not efficient, because they may report differences between two documents even if the documents have the same contents.

We can consider another approach; the approach that uses a database to record the histories. Roughly speaking, two research directions are related to this subject. One is *temporal databases*. A temporal database is a database that maintains past, present, and future data in the broadest sense. For the past decades, considerable research effort have been devoted to both of temporal relational databases [11] and temporal object-oriented

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 334–344, 2000.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE document SYSTEM "document.dtd">

<document>
  <title>XML</title>
  <authors>
    <author email="taro@db.aist-nara.ac.jp">Taro</author>
    <author>Jiro</author>
  </authors>
  <chapter title="Introduction">
    <section>
      XML stands for eXtensible Markup Language.
    </section>
  </chapter>
</document>
```

Fig. 1. An example of XML document

databases [10]. The other is repositories for XML documents, what is called *XML databases*. These studies intend to store XML documents into conventional database systems [3, 6, 8, 9]. If we combine techniques of these researches, we can preserve temporal aspects of XML documents. However, there are some disadvantages in this approach:

1. Temporal relational databases maintain data and temporal aspect of them in flat structures, whereas XML documents have tree structure.
2. On the other hand, temporal object-oriented databases can maintain tree structures by mapping a tree node (an element of the document) to an object, and an edge to a pointer, respectively. However, they keep temporal information inside of each object only, that is, we cannot utilize inference of temporal information among elements of the document.

In this paper, we propose a new data model for capturing histories of XML documents. This model is based on the XPath [15] data model, and extends it in some points: (1) edges have a label that represent their valid time; (2) string-value of text and attribute nodes are modeled as virtual nodes; and (3) text and attribute nodes can contain multiple string-value nodes. Furthermore, physical implementation models are proposed so that we can translate data represented in the model to XML documents. Since the formats are based on XML, any XML applications can process the resulting XML documents.

The rest of this paper is organized as follows. Section 2 introduces the XPath data model and its extension. Section 3 proposes physical implementations of our model. Section 4 shows related work, and Section 5 concludes this paper.

2 The Temporal XPath Data Model

2.1 The XPath Data Model

We adopt the XPath [15] data model as the basis of our data model. XPath is a language for addressing parts of an XML document, designed to be used by both XSLT [17] and

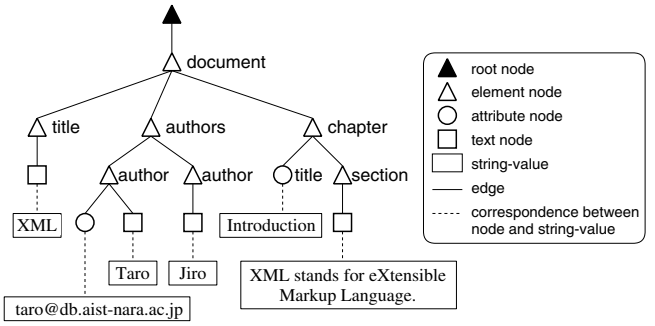


Fig. 2. XPath data model

XPointer [16], and operates an XML document as a tree. There are four kinds of nodes in the data model, namely, *root*, *element*, *text*, and *attribute* nodes[1]. The *root* node is the root of the tree. A root node does not occur except as the root of the tree. The *element* node for the document element is a unique child of the root node. There is an element node for every element in the document. The children of an element node are element and/or *text* nodes. Each element node has an associated set of *attribute* nodes. A text node never has an immediately following or preceding sibling that is a text node. For every type of node, we can determine a *string-value* for a node of that type. The formal definition of nodes are below.

Definition 1 (Node) *Let D be an XML document, $\boldsymbol{V}(D)$ is a set of nodes of D. There are four kinds of nodes, namely, root, element, text, and attribute nodes, each of which is refered to as r, $\boldsymbol{V_e}(D)$, $\boldsymbol{V_t}(D)$, and $\boldsymbol{V_a}(D)$, respectively. These sets are pairwise disjoint and $\boldsymbol{V}(D) = \{r\} \cup \boldsymbol{V_e}(D) \cup \boldsymbol{V_t}(D) \cup \boldsymbol{V_a}(D)$.*

Definition 2 (Edge) *$\boldsymbol{E}(D)$ is a set of edges (p,c) in an XML document D, where each pair is one of the followings: $p = r$, $c \in \boldsymbol{V_e}(D)$; or $p \in \boldsymbol{V_e}(D)$, $c \in \boldsymbol{V_e}(D)$; or $p \in \boldsymbol{V_e}(D)$, $c \in \boldsymbol{V_t}(D)$; or $p \in \boldsymbol{V_e}(D)$, $c \in \boldsymbol{V_a}(D)$; or $p \in \boldsymbol{V_a}(D)$, $c \in \boldsymbol{V_t}(D)$. These types are refered to as r-e, e-e, e-t, e-a, and a-t, respectively.*

Definition 3 (XML document) *An XML document is a 3-tuple $D = (\boldsymbol{V}(D), \boldsymbol{E}(D), r)$, where $\boldsymbol{V}(D)$ is a set of nodes; $\boldsymbol{E}(D)$ is a set of edges; and r is a special node in $\boldsymbol{V}(D)$ called* root *of the document.*

Figure 2 depicts the tree of the XML document in Figure 1 modeled by the XPath data model. In this figure, string-value of text and attribute nodes is illustrated with dashed line, whereas string-value of element nodes are omitted. This is because the string-value of an element node can be determined by concatenating string-value of its all descendants.

[1] For the sake of simplicity, we do not consider *namespace*, *processing instruction*, and *comment nodes*. Note that we can easily extend the following discussions to the original XPath data model.

2.2 Representation of Histories of XML documents

In this section, we will extend the XPath data model so that we can represent the history of XML documents. First of all, we introduce terminology from the research area of temporal databases [5, 11, 18].

The time domain We assume that there is one dimension of time, and we employ linear time model where time advances from the past to the future in a totally ordered fashion. In addition, we adopt discrete time model where time line consists of atomic *chronons*. Consecutive chronons may be grouped together into granules, with different groupings yielding distinct granularities. Time line is bounded by the relative beginning and the current time, represented by the special symbol "*now*." The current time is moving constantly as time goes by.

Time data types There are several temporal data types. The most basic one is a time *instant* which is a particular chronon on the time line. A *time intervals* $[t_s, t_e]$ is the time between two instants t_s and t_e where $t_s < t_e$, for example, [2000-01-01, 2000-01-31]. A *temporal element* $\{[t_1,t_2],[t_3,t_4],\ldots,[t_{n-1},t_n]\}$ is a finite set of time intervals. Incidentally, $[t_1 \ldots t_2, t_3 \ldots t_4, \ldots, t_{2n-1} \ldots t_{2n}]$ is an abbreviation of $\{[t_1,t_2],[t_3,t_4], \ldots,[t_{2n-1},t_{2n}]\}$.

Valid time The valid time of a fact is the time when the fact is true in the modeled reality. A fact may have associated any number of instants and intervals.

The Temporal XPath Data Model In the XPath data model, we can observe that: (1) modifications to an XML document is considered to be a sequence of the operations to the document tree: insertion, deletion, and replacement of an element or an attribute; (2) modifications to the content of a node can be considered as modifications to the node's text-node; and (3) an element can be considered as a container that holds children element, attribute, and text nodes, and hence, valid time of the node can be determined by computing the union[2] of all childrens' valid times.

From these observations, we extend the XPath data model so as to record the history of XML documents. The points are: (1) we distinguish every node's string-value as a virtual node; (2) we make attribute and text nodes possible to contain multiple string-value nodes; and (3) we introduce labeled edges so as to represent valid time of its descendants.

Definition 4 (String-value node) *Let D be an XML document,* $\boldsymbol{V_s}(D)$ *is a set of string-value node. Hence,* $\boldsymbol{V}(D) = \{r\} \cup \boldsymbol{V_e}(D) \cup \boldsymbol{V_t}(D) \cup \boldsymbol{V_a}(D) \cup \boldsymbol{V_s}(D)$.

Definition 5 (Edge) *Let D be an XML document,* $\boldsymbol{E}(D)$ *is a set of labeled edges* $((p,c),t)$*, where* (p,c) *is one of the followings:* $p = r$, $c \in \boldsymbol{V_e}(D))$*; or* $p \in \boldsymbol{V_e}(D))$, $c \in \boldsymbol{V_e}(D))$*; or* $p \in \boldsymbol{V_e}(D))$, $c \in \boldsymbol{V_t}(D))$*; or* $p \in \boldsymbol{V_e}(D))$, $c \in \boldsymbol{V_a}(D))$*; or* $p \in \boldsymbol{V_t}(D))$, $c \in \boldsymbol{V_s}(D))$*; or* $p \in \boldsymbol{V_a}(D))$, $c \in \boldsymbol{V_s}(D))$*. These types are referred to as r-e, e-e, e-t, e-a, t-s, and a-s, respectively. t is the label of the edge representing the valid time label of c with in terms of a temporal element. In other words, c exists for the time specified in t.*

[2] Let $t = [t_s,t_e]$ and $u = [u_s,u_e]$ be time intervals, union ($\cup$) of t and u is defined as $t \cup u = [min(t_s,u_s), max(t_e,u_e)]$.

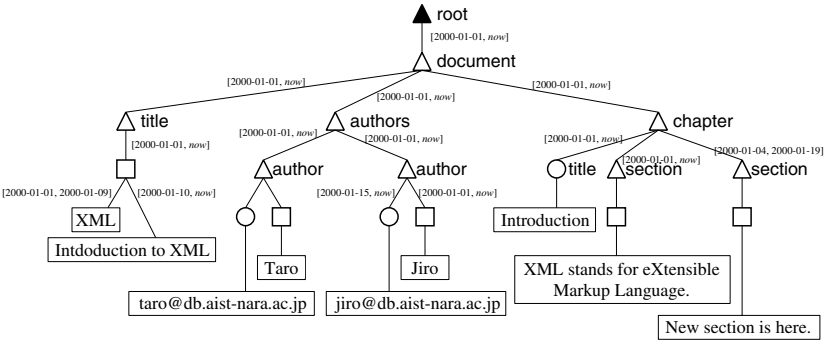


Fig. 3. Representation of histories

In the next, we give the definition of consistent temporal XML documents.

Definition 6 (Consistent temporal XML document) *A temporal XML document D is consistent if the following two conditions are satisfied: i) Let $v \in \boldsymbol{V}(D) - \{r\}$ be a node, p be the parent of v, t be the valid time label of the edge $((p,v),t)$, $v_1, v_2, \ldots, v_n$ be the children of v, and t_i be the valid time label of the edge $((v,v_i),t_i)$ $(1 \leq i \leq n)$. Then,*

$$\bigcup_{1 \leq i \leq n} t_i \subseteq t \ .$$

ii) Let r be the root of D, $c_1, c_2, \ldots, c_m$ be the children of r, u_i be the valid time label of the edge $((r,c_i),u_i)$ $(1 \leq i \leq m)$. Then,

$$u_i \cap u_j = \emptyset$$

where $i \neq j$ and $1 \leq i, j \leq m$.

Figure 3 shows an example of temporal XML document for the XML document in Figure 1. In this figure, for the sake of readability, valid time labels are omitted when those values are identical to their closest ancestors.

Inferencing Valid Time Labels Practically, it is troublesome to consistently specify all valid time labels in a document tree. However, we do not need to do this, that is, we can infer entire valid time labels from a part of them if sufficient information is supplied.

Proposition 1 *Let D be a temporal XML document, $r \in \boldsymbol{V}(D)$ be the root node of D, $S \subset \boldsymbol{V}(D)$, and $\overline{S} = \boldsymbol{V}(D) - S$, all valid time labels in D can be derived at least all valid time labels of a cutset* [3] *$[S, \overline{S}]$ where $r \in S$ and all leaf nodes in D are contained in $\overline{S}$.*

(Proof) *We give a way to infer valid time labels that satisfies the conditions in Definition 6. Let $l_1, l_2, \ldots, l_n$ be the leaf nodes, there is a path from the root r to l_i, $\langle r, e_1, v_1, e_2, v_2, \ldots, v_{m-1}, e_m, l_i \rangle$, since D is a tree. There exists an edge $e_j (1 \leq j \leq m)$ such that $e_j \in [S, \overline{S}]$*

[3] Let G be a graph, E and V be the sets of edges and nodes in G. A cutset of G is a proper subset of E which divide V into S and $\overline{S}(= V - S)$, and denoted by $[S, \overline{S}]$.

from the definition, and we can obtain the time label t_j of the edge e_j consequently. Then we set the valid time labels $e_{j+1}, e_{j+2}, \ldots, e_m$ to t_j. After doing this for each leaf node, we can infer the rest of valid time labels by computing union of all time labels connected to a node in a bottom-up manner.

Restoring Past States It is easy to compute a past state of XML documents using our data model; all we have to do is that we recursively prune edges that are not available at specified time, and remove labels from edges. From the definition 5, this process eliminates duplications of string-value nodes in text and attribute nodes, and thus, obtained document tree become conforming to the XPath data model.

3 Implementation

This section describes issues on the implementation of our data model. At first, we investigate operations of our data model. In the next, we introduce physical implementation models for translating documents in our model into XML documents.

3.1 Operations

Generally speaking, there are many ways of accessing XML documents, but those methods can roughly be categorized into two methods: one is that we use specialized APIs for XML documents such as DOM, and the other is that we directly edit the documents using application programs like editors. For the former, we can implement our data model by extending the DOM API. The latter is relatively difficult, since the system must find the differences between the original documents and modified ones by itself. However, some application programs have been developed for this purpose [4], and thus we can assume that the differences are given. After that we can use DOM API. For this reason, we only consider extensions to the DOM API.

Operations based on DOM API Here we only mention extensions to the original DOM API.
Operations on Nodes:

- insertBefore(Node newChild, Node refChild), appendChild(Node newChild)
  No extension is required.
- removeChild(Node oldChild)
  As described in Section 2.2, all we have to do is to replace *now* in the valid time to the current time stamp.
- replaceChild(Node newChild, Node oldChild)
  We call insertBefore(newChild) in the first, and then, call removeChild(oldChild).

Operations on CharacterData: Note that operations on CharacterData correspond to that on string-value nodes in our data model.

- setData(String data), appendData(String arg)
  If string-value node is *null*, that is, the target element is an empty element, we have to create the node, and then, call the original setData operation. Otherwise, no extension is required.
- insertData(int offset, String arg), deleteData(int offset, int count), replaceData(int offset, int count, String arg)
  No extension is required.

Operations on Elements: Operations on elements also include interfaces to attributes.

- removeAttribute(String name), removeAttributeNode(Attr oldAttr)
  We replace *now* in the valid time to the current time stamp like the case of removeChild.
- setAttribute(String name, String value), setAttributeNode(Attr newAttr)
  These operations are for creating new attributes or rewriting contents of attributes. In the latter case, that is, when the attribute named *name* already exists, we call removeAttribute(name) at first. Then call setAttribute(name, value).

3.2 Physical Implementation

Here we discuss how to implement documents in our model in XML documents. There are two distinct directions; one is for implementing our data model to XML documents as they are, and the other is for implementing our data model retaining the original form of XML documents. We call the former implementation *full*, and the latter *simplified*. For both implementations, we newly introduce some tags and attributes in order to represent temporal information. In that case, we use Namespaces in XML Recommendation [14] to avoid conflicts of tag and attributes.

Full Implementation In full implementation, we straightforwardly map every nodes and valid time labels into separated elements and attributes. To this end, the following three items are introduced: (1) *valid* attribute, (2) *attribute* node, and (3) *string-value* node. Correspondence between our model and these items are:

Valid time labels A valid time label is mapped to a attribute *valid*. Thus, every element has one attribute *valid* in this implementation.

Attribute nodes An attribute node is mapped to an *attribute* element. The reason why we do not map it to an actual attribute is that it may have multiple string-value nodes over time. Full implementation represents this situation by including multiple *stringvalue* elements, described below, in an attribute element.

String-value nodes A string-value node is mapped to an element *stringvalue*.

Figure 4 represents a full implementation of the document in Figure 3. As we can see from this figure, although full implementation is faithful to our data model, the representation is complicated and sometimes redundant, and thus, dissimilar to the source document. Hence, we can consider some optimization to this implementation.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE document SYSTEM "document.dtd">

<document xmlns:time="http://db-www.aist-nara.ac.jp/time"
        xmlns="http://db-www.aist-nara.ac.jp/document"
        time:valid="2000-01-01, now">
  <title time:valid="2000-01-01, now">
    <time:stringvalue time:valid="2000-01-01, 2000-01-09">
      XML
    </time:stringvalue>
    <time:stringvalue time:valid="2000-01-10, now">
      Introduction to XML
    </time:stringvalue>
  </title>
  <authors time:valid="2000-01-10, now">
    <author time:valid="2000-01-01, now">
      <time:stringvalue time:valid="2000-01-10, now">
        Taro
      </time:stringvalue>
      <time:attribute name="email" time:valid="2000-01-10, now">
        taro@db.aist-nara.ac.jp
      </time:attribute>
    </author>
        :
```

Fig. 4. Full implementation

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE document SYSTEM "document.dtd">

<document xmlns:time="http://db-www.aist-nara.ac.jp/Time"
        xmlns="http://db-www.aist-nara.ac.jp/Document"
        time:valid="2000-01-01, now">
  <title time:valid="2000-01-01, 2000-01-09">XML</title>
  <title time:valid="2000-01-10, now">Introduction to XML</title>
  <authors time:valid="2000-01-01, now">
    <author email="taro@db.aist-nara.ac.jp"
            time:valid="2000-01-01, now">
      Taro
    </author>
    <author time:valid="2000-01-01, 2000-01-14">Jiro</author>
    <author email="jiro@db.aist-nara.ac.jp"
            time:valid="2000-01-15, now">
      Jiro
    </author>
  </authors>
        :
```

Fig. 5. Simplified implementation

Simplified Implementation Although full implementation captures all information of the data model, the description is redundant and is quite different from its original document. For this reason, we introduce simplified implementation by which we can retain original form of documents in comparison with the full implementation. In this implementation, we only use *valid* attribute, and represent the history of a document by duplicating elements which are updated. Figure 5 shows a simplified implementation of the document in Figure 3.

The following procedure *mapElement* is the algorithm for implementing an XML document in our model using simplified implementation. The procedure takes three arguments, namely, an element e in our data model, start and end time points t_s, t_e, and outputs e' as the result in which e is translated in XML.

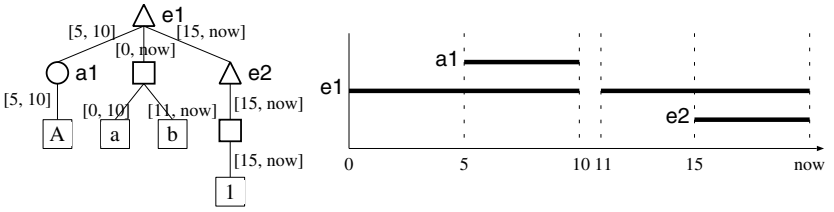


Fig. 6. mapElement

Procedure **mapElement**(e, t_s, t_e)

1. Check if e contains text or attribute nodes which have two or more string-value nodes.
2. In the case that 1 is true:
   (a) Retrieve valid time intervals $[t1_s, t1_e], [t2_s, t2_e], \ldots, [tn_s, tn_e]$ from the string-value nodes, extract all of the start and end time points $T = \{t1_s, t1_e\} \cup \{t2_s, t2_e\} \cup \ldots \cup \{tn_s, tn_e\}$ and sort T.
   (b) Extract all possible maximal non-overlapping time intervals from T, for example, we can extract three time intervals $\{[1,4],[4,7],[7,now]\}$ from $T = \{1,4,7,now\}$, and let the result be U.
   (c) For each $u = [u_s, u_e] \in U$, create an element e' that includes (1) u as its *valid* attribute, and (2) attributes and a string-value as of u.
   (d) For each child element c in e, invoke **mapElement**(c, u_s, u_e) with each $u \in U$.
3. Otherwise
   (a) Create an element e' that includes e's valid time, attributes, and string-values.
   (b) Let $u = [u_s, u_e]$ be e's valid time. For each child c in e, invoke **mapElement** (c, u_s, u_e).

Figure 6 shows an example. For the element `e1`, the first step is true, then the step two and three calculates the number of changes in `e1` and its valid time. The chart in Figure 6 shows the intuitive explanation; `e1` has changed for two times in total, and hence, we create three `e1` elements like below:

```
<e1 time:valid="0, 5">a</e1>
<e1 a1="A" time:valid="5, 10">a</e1>
<e1 time:valid="11, now">b</e1>
```

At last, we call **mapElement** for `e2` with each valid time in `e1`, and we obtain:

```
<e1 time:valid="0, 5">a</e1>
<e1 a1="A" time:valid="5, 10">a</e1>
<e1 time:valid="11, now">
  b
  <e2 time:valid="15, now">1</e2>
</e1>
```

Querying Temporal XML Documents Roughly speaking, there are two ways when querying temporal XML documents. In the first case, we obtain a snapshot XML document at first, then we query the snapshot using an ordinary (non-temporal) query language. In the second case, we use a query language specialized for our temporal XML documents. In the latter case, we need to develop a specialized query language, whereas we do not need any extension to existing technologies in the former case. Development of query language for our model is one of our future work.

Even if we do not use query language, we can process temporal queries in an ad-hoc approach using XPath. Note that we assume that the function *during(a, b)* that tests time interval *b* is contained by *a*.

Example 1 *Query: Find the title as of "2000-01-15."*

```
//title[during(@time:valid, "2000-01-15")]
```

Example 2 *Query: Find the time when Jiro obtained his email address "jiro@db.aist-nara.ac.jp."*

```
//author[@email = "jiro@db.aist-nara.ac.jp"]/@time:valid
```

4 Related Work

Chawathe et al. investigated a model for representing changes in semistructured data and a language for querying over these changes [2]. They proposed DOEM, which is a temporal extension of OEM [7], and query language named *Chorel* which is a query language for DOEM. In particular, they represented changes in OEM by introduction of annotations which contain *basic change operations*. In other words, DOEM captures the history of a semistructured data as a sequence of change operations.

Bertino et al. studied a formalization of navigational accesses in a temporal object model [1]. In this study, they formally defined the notion of temporal path expression, and gave a set of conditions ensuring that an expression always results in a correct access at runtime. Although we do not need to consider type checking, we have to establish the formal basis that ensures the correctness of expressions in our model.

WebDAV (Web-based Distributed Authoring and Versioning) [12] is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers. In this framework, we can specify metadata of a Web document in a sequence of well-formed XML document. Taking this feature, temporal information of our data model can be embedded in Web documents. This may quite useful to capture the histories of editorial process of Web documents, since current WebDAV do not take into account such information.

5 Conclusions

In this paper, we proposed a data model for representing temporal XML documents. The model, which is based on XPath, is capable of representing the whole history of an XML document. Furthermore, we investigated several issues on implementation of our model, such as, DOM API based operations, physical implementations, and temporal query expressions based on XPath. Future work includes implementation of our data model. Currently, we are planning to implement the model on an object-relational database.

Acknowledgements

This paper has benefited from useful discussions with Yohei Yamamoto. Takeshi Sannomiya gave us many valuable comments. This work was supported in part by a grant from the Japan Society for the Promotion of Science (No. 11480088, 12680417).

References

1. E. Bertino, E. Ferrari, and G. Guerrini. Navigational accesses in a temporal object model. *IEEE TKDE*, 10(4):656–665, 1998.
2. S. S. Chawathe, S. Abiteboul, and J. Widom. Representing and querying changes in semistructured data. In *Proc. of ICDE*, pages 4–13, 1998.
3. D. Florescu and D. Kossmann. Storing and querying XML data using an RDMBS. *IEEE Data Engineering Bulletin*, 22(3):27–34, 1999.
4. IBM alphaWorks. XML Diff and Merge Tool. http://www.alphaworks.ibm.com/formula/xmldiffmerge.
5. C. S. Jensen and C. Dyreson (editors). The Consensus Glossary of Temporal Database Concepts — February 1998 Version. http://www.cs.auc.dk/ csj/Glossary/, 1998.
6. Oracle Corporation. Using XML in Oracle Database Applications Part 2: About Oracle XML Products. http://technet.oracle.com/tech/xml/info/index2.htm?Info&htdocs/otnwp/about_oracle_xml_products.htm, Nov. 1999.
7. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proc. of ICDE*, pages 251–260, Mar. 1995.
8. J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, Jeffrey, and F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *Proc. of VLDB Conf.*, pages 302–314, Sept. 1999.
9. T. Shimura, M. Yoshikawa, and S. Uemura. Storage and retrieval of XML documents using object-relational databases. In *Proc. of DEXA 1999*, pages 206–217, Sept. 1999.
10. R. T. Snodgrass. *Modern Database Systems: The Object Model, Interoperability, and Beyond*, chapter 19. Addison-Wesley / ACM Press, 1995.
11. A. U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases: Theory, Design, and Implementation*. The Benjamin/Cummings Publishing Company, Inc., 1993.
12. J. Whitehead and M. Wiggins. WEBDAV: IETF standard for collaborative authoring on the web. *IEEE Internet Computing*, pages 34–40, September/October 1998.
13. World Wide Web Consortium. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml. W3C Recommendation 10 February, 1998.
14. World Wide Web Consortium. Namespaces in XML. http://www.w3.org/TR/REC-xml-names/. W3C Recommendation 14 January 1999.
15. World Wide Web Consortium. XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath/. W3C Recommendation 16 November, 1999.
16. World Wide Web Consortium. XML Pointer Language (XPointer). http://www.w3.org/TR/xptr/. W3C Working Draft 6 December, 1999.
17. World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt/. W3C Recommendation 16 November, 1999.
18. C. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian, and R. Zicari. *Advanced Database Systems*, chapter 5. Morgan Kaufmann Publishers, Inc., 1997.

Blind Queries to XML Data

Ernesto Damiani[1] and Letizia Tanca[2]

[1] Universitá di Milano and George Mason University, Computer Science Dept.
4400 University Drive, Fairfax, VA 22030, US
edamiani@cs.gmu.edu
[2] Politecnico di Milano, Dipartimento di Elettronica e Informazione
Via Ponzio 1, 20100 Milano, Italy
tanca@elet.polimi.it

Abstract. A flexible query model is presented for *well-formed* XML documents. Our approach relies on modeling XML documents as labeled graphs and selectively extending their structure by computing fuzzy estimates of the *importance* of the information they provide, at the granularity of XML tags. The result of such an extension is a *fuzzy labeled graph*. Query results are subgraphs of this fuzzy labeled graph, presented as a *ranked list* according to their degree of matching to the user query.

1 Introduction and Motivations

XML (eXtensible Markup Language) is a markup metalanguage designed to enable semantics-aware tagging of World Wide Web information [XML]. Generally speaking, an XML document is composed of a sequence of nested elements, each delimited by a pair of start and end tags (e.g., `<tag>` and `</tag>`). An XML document is *well-formed* if it obeys the basic syntax of XML (e.g., non-empty tags are properly nested, each non-empty start tag have the corresponding end tag). Well-formed documents are also *valid* if they conform to a proper *Document Type Definition*(DTD). A DTD is a file which contains declarations for *elements* (i.e. tags), *attributes*, *entities*, and *notations* that will appear in XML documents. DTDs state what names can be used for element types, where they may occur, how each element relates to the others, and what attributes and sub-elements each element may have. Attribute declarations in DTDs specify the attributes of each element, indicating their name, type, and, possibly, default value. Entities and notations are of paramount importance in the description of physical XML documents, but will not be considered in this paper, as we focus the analysis on the XML documents' logical structure. However useful, DTDs have suffered from several problems. For instance, DTDs are extremely difficult to read or author, because the syntax of a DTD is not XML, but rather a DTD-specific grammar that is similar to (but still different from) XML. Moreover, validation is time-consuming and does not provide type-checking. Finally, DTDs do not deal *scoping* and *namespaces*, which makes them unusable in many application scenarios. For the above reasons, many XML-based systems simply define their own type information representations as XML *vocabularies* [Mic99]. Most current XML query languages assume the structure of target XML documents to

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 345–356, 2000.

be known to the user [Cer99b], [Clu99]. As a result, such languages are awkward to use for querying well-formed XML data. The tags in well-formed XML documents carry information about data semantics that should however be exploited in query execution. In the sequel, we shall focus on flexible techniques for posing *blind* queries to well-formed XML information, representing well-formed XML documents and queries as *labeled directed graphs* $G = (V, E, L, f, g)$, whose node set V comprises both nodes representing tags and nodes representing text/multimedia content and attributes. Arcs belonging to $E \subseteq V \times V$ may represent, according to their labelling (given as usual by a function $f : E \rightarrow L$, where $L = \{e - contains, a - contains, link, id - idref\}$ is a set of *relation labels*) tag and attribute inclusion, hypertext links, and `ID-IDREF` relationships. Another function $g : V \rightarrow I^*$ (where I^* is the set of strings built over a suitable alphabet I) represent the value or content associated to a terminal element or attribute. The sub-graph representing (element and attribute) containment alone is in most cases a tree, where leaf nodes represent content and values, while non-leaf nodes correspond to tags. While throughout the paper we shall, for the sake of simplicity, refer to such sub-graph, we remark that our approach can be straightforwardly applied to links and `ID-IDREF` relationships as well. In this setting a query can be, without loss of generality, represented as a *graph pattern*: query execution involves finding a match of the pattern inside *document graphs* representing XML documents. Graph-based representations have been widely used in the framework of DTD-based XML query languages [Cer99a], [Clu99] as well as for a variety of XML-related environments and tools. In order to enhance flexibility we shall provide a different query execution model for blind queries, which does *not* rely on the straightforward computation of pattern matching between the document and the query; rather, we shall find this matching after a number of preliminary steps, with the aim of narrowing the gap between query and document structures. The rationale for this approach is that even without a DTD, target XML documents can be used to estimate the intended importance of XML elements as perceived by the document designer. Moreover, query structure identifies the importance of XML elements as perceived by the user. Our approach relies on three basic steps:

1. Weighing the target document content on the basis of the document's topological structure, and tag repertoire. The output of this step is a *fuzzy labeled graph* [Cha92].
2. Transformation of the fuzzy labeled graph. In this step, also carried out at document design time, the closure of weighed documents is computed. At query execution time, the result is tailored performing an α-cut operation on the basis of a threshold parameter provided by the user. The output of this step is a new, tailored target graph.
3. Computation of a similarity matching between the subgraphs of the tailored document and the query graph, according to the type of matching expressing the query semantics selected by the user.

Our technique offers the choice between different notions of similarity between the query graph and document subgraphs. Moreover, similarity is computed be-

```
<department>
   <dptname> Web Technology department <dptname>
   <division name = "R&D" >
      <group name = "XML Applications" >
         <research>
            <description>
               The group's research activity is in the area
               of markup languages
            </description>
            <contact>
               <address> Cherry Ave </address>
               <e-mail> xml@acme.com </email>
            </contact>
         </research>
         <members>
            <person>
               <flname> Sam Dale </flname>
               <address> 1st Ave </address>
            </person>
         </members>
      </group>
   </division>
</department>
```

Fig. 1. A well-formed XML document

tween fuzzy rather than crisp graphs, providing a degree of matching [Gol96] that will be used to rank results. Fig. 1 shows a sample XML document that will be used throughout the paper. Fig.2 shows the graph corresponding to the document of Fig. 1. Here we do not adopt the notation of a specific language or environment; rather, we represent non-terminal elements as rectangles, attributes and terminal elements as ellipses (for the sake of clarity, terminal elements are drawn thicker than attributes). Since document nodes are unique, we shall specify a numerical *Object ID*, (`OID`) for each node. This causes no loss of generality, since such a `OID` can be easily computed based on the unique path reaching the element on the containment tree. Since our sample document only features containment links, labels on arcs have been omitted in Fig.2, while attributes' value and terminal elements' content are depicted inside corresponding nodes. We are now ready to define the general concept of a *query* to XML information. Of course, standard text retrieval techniques could be used to search for tags (such as, in our example, `<department>` and `<address>`) as well as for their desired content. Standard *Boolean* techniques for text retrieval search rely on a lexicon, i.e. a set of terms $r_1, r_2, , r_k$ and model each document as a Boolean vector of length k, whose i-th entry is `true` if r_i belongs to the document. In this setting, a query is simply a Boolean expression (e.g., a conjunction) whose operands are terms or stems (possibly including *wildcards*), and its result is the set of documents where the Boolean expression evaluates to `true`. Thus, document ranking is not supported in a pure Boolean setting. Fuzzy and probabilistic techniques have been proposed to overcome this problem [Bue81], whose result are usually *ranked lists* of documents.These techniques are currently in use for search engines dealing with HTML documents, and could of course be employed for XML data as well, at the price of loosing the information conveyed by the document's structure. Recently, more sophisticated approaches (e.g. algebraic

ones, [Cla95]) have been introduced, leading to XML processing languages such as XQL [Rob99], which also provides search capabilities.

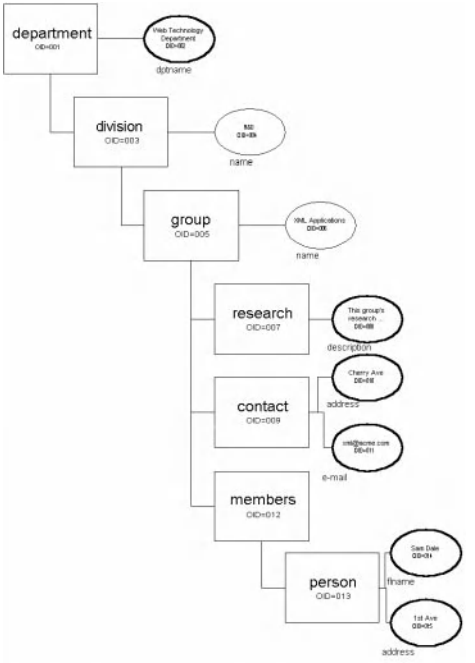


Fig. 2. A Sample XML graph

The database community has proposed several fully-fledged query languages for XML, some of them as a development of previous languages for querying semi-structured data; two detailed comparisons (both involving four languages) can be found in [Cer99b] and [Clu99] while many preliminary contributions and position papers about XML querying are collected in [QL98]. Here, we shall not attempt to describe such languages in detail; rather, we only refer to the common features of XML-QL [Deu99], YaTL [Clu98] and XML-GL [Cer99a]. Two features shared by these languages [Clu99] are relevant to our discussion:
User-provided patterns, based on the assumption that the user is aware enough of the target document structure to be able to formulate a pattern (a path, or more generally a graph pattern) that can be matched against the target XML documents for locating the desired information. Flexibility support is obtained by means of wild cards [Clu99].
Set-oriented query result: all query languages retrieve portions of XML documents, namely the ones matching the user-provided pattern. All retrieved portions equally belong to the query result set, even when the query exploits the

facilities provided by the language for partial or flexible pattern matching.
When querying well-formed XML information, the assumption that the user is aware of the target document structure is debatable, because users cannot exploit a DTD as a basis for the query graph's structure. Often, all users can rely on is a sample document, or at most a tag repertoire, i.e. the XML *vocabulary* used throughout the XML document base. In this situation, trying to find a match of the query pattern to a part of the target document is likely to result in *silence*, as the query topology will generally have little to do with the document's structure. Fig.3 shows a simple "blind" query composed on the basis of the vocabulary of the document in Fig.1. The user is interested in finding contact information for the departments, but has no clue on the target document structure, and searching for a match of the query pattern inside the document would result in a failure. While path wild cards may sometimes alleviate this problem, they cannot fully eliminate it: if there is no path *of inclusion arcs* in the target document graph connecting two nodes specified as connected in the blind query pattern, such pattern will not match even if the two nodes are indeed reachable from each other in the target documents following a path including hypertext or `ID-IDREF` links. We also remark that in our opinion queries like the one in Fig.3 do not dictate the exact structure of the query result; rather, they provide a loose example of the information the user is interested in. Therefore, several degrees of matching should be possible. This is the usual situation in the field of multimedia databases [Fag96] where query results are ranked lists according to a similarity measure.

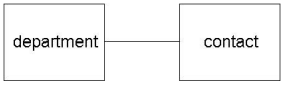


Fig. 3. A "blind" query

2 Weighing XML Information

Intuitively, we need to match the query graph against the document after extending the document's graph in order to bypass links and intermediate elements which are "not relevant" from the user's point of view. In order to perform the extension in a sensible way, we shall first evaluate the *importance* of well-formed XML information at the granularity of XML elements. We rely on *fuzzy weights* to express the *relative importance* [Bos98] of information at the granularity of XML elements. Low values will correspond to a negligible amount of information, while a value of 1 means that the information provided by the element (including its position in the document graph)is extremely important according to the document author. Other than that, the semantics of weights is only

defined in relation to other weights in the same document/query. Two main approaches can be used to compute automatically an estimate of elements' importance: *structure-related* and *tag-related* document weighing.

Structure-related weights Structure-related weighing weighs the arcs of an XML graph using topological parameters related to the position of XML elements and attributes. This is obtained by computing a function $w_{arc} : E \rightarrow [0, 1]$ estimating the importance of the arc. This function associates in a natural way a fuzzy value to the arc. Topological parameters to be considered include *nesting*, i.e. the length of the path to the terminal element of the arc from the document root node, and *fan-out*, i.e. the number of elements/attributes directly contained in the terminal element of the arc under consideration. Structure-related weighing can readily be applied to both documents and queries; however, a basic distinction should be drawn. When weighing a query, topological parameters estimate the importance of a generic XML element as perceived by the user. On the other hand, document weights are estimates of the importance of individual tags, as perceived by the document designer. Though we shall deal with document weighing only in this paper, our techniques are readily extendable to take into account weighted queries as well.

Tag-related weights This technique labels the nodes of an XML graph with their relative importance, by means of a function $w_{node} : V \rightarrow [0, 1]$. Tag-related weights could be obtained by polling the user; alternatively, they can be frequency-related. The latter notion associates tag importance with the frequency of XSL-like *path expressions* [XSL] ending with that tag, throughout the document base. A path expression uniquely identifies a tag in a given position inside an XML document; for instance, `department:group:contact:address[Cherry Ave]` uniquely identifies the first `<address>` tag in Fig.1. We are now ready to outline the actual computation of the fuzzy weights using the structure-related technique. We use a function $w_{arc} : E \rightarrow [0, 1]$ to weigh each arc (n_i, n_j) of the target document graph G. An example of such function is, for instance, the *normalized distance* from root, defined as follows:

$$w_{arc}(n_i, n_j) = \frac{d_{max} - l}{d_{max}} \tag{1}$$

where d_{max} is the length of the longest path starting from a root node and l is the distance from n_j to the root. This function establishes a simple inverse relation between tags importance (as perceived by the document designer) and their nesting level. Another suitable function associates to each arc (n_i, n_j) in G the *normalized cardinality* of the sub-tree G' (obtained taking inclusion arcs only into account) whose root is n_j, namely

$$w_{arc}(n_i, n_j) = \frac{\mid G'(n_j) \mid}{\mid G \mid} \tag{2}$$

This weighing function (whose value does not depend on n_i) is non-monotonic w.r.t distance from root and estimates each tag's importance via the size of the subtree rooted in it. Applying the above weighing procedure to the well-formed document in Fig.1, and using the function of Eq.(1) we obtain Tab. 2.

start node	end node	weight
department	name	4/5
department	division	4/5
division	name	3/5
division	group	3/5
group	name	2/5
group	members	2/5
group	research	2/5
group	contact	2/5
members	person	1/5
research	descr	1/5
contact	addr	1/5
contact	e-mail	1/5
person	flname	0
person	address	0

Table 1. Structure-related weights for the sample document in Fig.1

Of course, no a priori definition of weighing function will be satisfactory in all cases; however, the obvious solution of polling the document designer for each XML document to be weighed is not feasible in many cases, as documents are often generated automatically. Alternatively, the document designer could be polled for designating top importance tags in the XML vocabulary, and the simple function of Eq.(1) could be applied to documents using these nodes as starting points. A sample tag-related weighing for the document in Fig. 1 is given in Tab. 2. In both our weighing models, weights do not depend upon the content/value of the XML element or attribute involved; this is indeed a drawback which limits the semantics of weights. For instance, an XML element such as `<PRICE>` could be considered important only when its content lies inside a given range of values [Dub99]. In principle, this problem could be solved introducing an additional dependency [Dub88] between weights and the content/value of the corresponding element/attribute. Checking this additional dependency is however bound to be computationally very expensive.

3 Fuzzy Closure Computation

Once the weighing is completed, the *fuzzy closure* C of the fuzzy labeled graph is computed. Intuitively, computing graph-theoretical closure entails inserting a new arc between two nodes if they are connected via a path of any length in the original graph. Computing the closure is well-known to be polynomial in the number of nodes of the graph. In our model, the weight of each closure arc in $C - G$ is computed by aggregating via a function T the weights of the arcs belonging to the path it corresponds to in the original graph. Namely, for each arc n_i, n_j in the closure graph C we write:

$$w_{arc}(n_i, n_j) = min_T\{w_{arc}(n_i, n_r), w_{arc}(n_r, n_s), ..., w_{arc}(n_t, n_j)\} \qquad (3)$$

node	weight
department	1
name	1
division	1
name	1
group	1
name	1
members	0.8
research	1
contact	0.8
descr	1
addr	0.8
e-mail	0.6
person	0.7
flname	0.7
address	0.7

Table 2. Sample tag-related weights for the sample document in Fig.1

where $\{(n_i, n_r)(n_r, n_s), ..., (n_t, n_j)\}$ is the set of the shortest paths from n_i to n_j in G and T is a standard triangular t-norm [Kli88], i.e. a binary function $[0,1]^2 \rightarrow [0,1]$ enjoying the $\wedge$-conservation property $(T(0,0) = 0; T(x,1) = T(1,x) = x)$, monotonicity $()x_1 \leq x_1' \wedge x_2 \leq x_2' \rightarrow T(x_1, x_2) \leq T(x_1', x_2'))$, commutativity $(T(x_1, x_2) = T(x_2, x_1))$and associativity $(T(T(x_1, x_2), x_3) = T(x_1, T(x_2, x_3)))$. Among functions satisfying these axioms, *min* is also the only idempotent one which well preserves query optimization properties [Fag96]. Other monotonic norms, such as average-based ones, do not exhibit $\wedge$-conservation, but promote a more *utilitaristic* view where the higher value of importance of an element can often *positively compensate* for a lower value of another one. In other words, it may happen that $T(x,y) \geq min(x,y)$. For a complete repertoire of t-norms, including non-triangular ones, see [Gup91]. The technique introduced above can be extended to include tag-related weights by plugging modified weights w'_{arc} in Eq.(3) obtaining $w'_{arc}(n_i, n_r) = T(w_{node}(n_r), w_{arc}(n_i, n_r))$. In this case, when the norm $T = min$, $w_{node}(n_r) = 0$ means that the arc (n_i, n_r) must be ignored in the computation whatever the value of $w_{arc}(n_i, n_r)$ for all nodes n_i connected to it. Intuitively, the closure computation step computes an extended structure of the document. Selecting the type of t-norm to be used for combining weights means deciding if and how a low weight on an intermediate element should affect the importance of a nested high-weight element. For instance, suppose a node n_j is connected to the root via a single path of length 2, namely $(n_{root}, n_i)(n_i, n_j)$. If $w_{arc}(n_{root}, n_i) << w_{arc}(n_i, n_j)$ the weight of the closure arc (n_{root}, n_j) will depend on how the t-norm T combines the two weights. In other words, how much should the high weight of (n_i, n_j) depreciate because the arc is preceded by (comparatively) low-weight one (n_{root}, n_i) ?. The conservative choice $T = min$ does not always agree with humans' intuition, because the *min* operator gives a value that depends only on one of the operands without considering the other

[Dub96] (for instance, we have the *absorption property*: $T(x, 0) = 0$). Moreover, it does not provide the *strict-monotonicity* property ($\forall y, x' > x \rightarrow T(x', y) > T(x, y)$). In other words, an increase in one of the operands does not ensure the result to increase if the other operand does not increase as well). To understand the effect of the *min*'s *single operand dependency* in our case, consider the two arc pairs shown below:

1. (<department><division>0.2)(<division><group>0.9)
2. (<department><division>0.3)(<division><group>0.4)

when the *min* operation is used for conjunction, arc pair (2) is ranked above arc (1), while most people would probably decide that arc pair (1), whose second element has much higher importance, should be ranked first. The other t-operators have the following common properties [Kli88]:

$$x = 1 \vee x = 0 \vee y = 1 \vee y = 0 \rightarrow T(x, y) = x \vee T(x, y) = y \tag{4}$$

$$T(x, y) \leq min(x, y) \tag{5}$$

Property 5 warns us that, while the other t-norms somewhat alleviate the single operand dependency problem of the *min* for arc pairs (using the product, for instance, the outcome of the previous example would be reversed), they may introduce other problems for longer paths. Let's consider the following example:

1. (<department><division>0.1)(<division><group>0.9)(<group><name>0.1)
2. (<department><division>0.2)(<division><group>0.5)(<group><name>0.2)

In this case we get $T(x, y, z) = T(x, T(y, z)) = 0.009$ for the first path, while the second gets 0.02; again this estimate of importance that ranks path (2) above path (1) may not fully agree with users' intuition. Tab.3 shows the elements' weights for the closure graph when T is the arithmetic average. The third column contains the final weight taking tag-related weights of Tab. 2 into account.

4 Query Execution

We are now ready to outline our query execution technique. First, we weigh the target document graph G and of the query graph Q according to structure-related or tag-related techniques. Weights on target documents can be computed once for all (in most cases, at the cost of a visit to the document tree). Secondly, we compute the closure graph C of G using a fuzzy or product-based conjunction of the weights. This operation is dominated by matrix multiplication, and its complexity lies in between $O(n^2)$ and $O(n^3)$ where n is the cardinality of the node-set V of the target document graph. Again, graph closure can be precomputed once for all and cached for future requests.
Then, we perform a *cut* operation on C using as a threshold (this operation

start node	end node	arc weight	final weight
dept	group	0.7	0.85
dept	research	0.6	0.8
dept	description	0.5	0.75
dept	contact	0.6	0.7
dept	members	0.6	0.7
dept	person	0.5	0.65
division	research	0.5	0.75
division	contact	0.5	0.65
division	members	0.5	0.65
division	person	0.4	0.55
group	person	0.3	0.5

Table 3. The weights of the closure graph when T is the arithmetic average

gives a new, tailored target graph TG). The cut operation simply deletes the closure arcs whose weight is below a user-provided threshold α, and is linear in the cardinality of the edge-set of $C - G$.

Finally, a fuzzy similarity matching is computed between the subgraphs TG of the tailored document and the query graph Q, according to selected type of matching. This operation coincides with the usual query execution procedure of pattern-based query languages, and its complexity can be exponential or polynomial w.r.t the cardinality of the node-set V of the target document graph [Com98], depending on the allowed topology for queries and documents [Coh93]. The first steps of the above procedure are reasonably fast (as document weights and closure can be pre-computed, required on-line operation consists in a sequence of one-step lookups). The last step coincides with standard pattern matching in the query execution of XML query languages [Cer99a], and its complexity clearly dominates the other steps. To allow for maximum flexibility, several notions of matching can be employed for locating the fuzzy subgraphs of the extended document graph and computing their degree of matching with respect to the user query. In the case of *graph embedding*, matching between document subgraphs and the query graph is defined as a function φ associating query nodes to document nodes in such a way that edges and labels are preserved.

If *graph isomorphism* is required, matching between document subgraphs is a function φ as above, which in this case must be a one-to-one mapping.

Our execution procedure consists of three steps. First of all, given the query $Q = (V, E, f)$, without taking membership values into account, we locate a matching subgraph $G' = (V', E', f')$ in the extended document graph (using, for instance, crisp depth first search), such that that there is a mapping $\varphi : V \rightarrow V'$ preserving arcs and arc labels. [1]. A procedure `FindMatch` is used, according to the desired type of matching; in the case of graph embedding, its complexity

[1] Moreover, this matching ensures that if values are specified on terminal nodes in the query graph, they also must appear as content labels of the corresponding nodes in the input document graph

is polynomial in $\mid V \mid$ for simple queries [Com98]. Secondly, we compute the *ranking function* $J(Q, G')$ as follows:

$$J = \wedge_{n_i,n_j \in E} w_{arc}(\varphi(n_i), \varphi(n_j)) = T(w_{arc}(\varphi(n_i), \varphi(n_j))...) \tag{6}$$

. In the second part of Eq.(6), we straightforwardly use t-norm associativity to compute the conjunction T over all edges $\varphi(n_i), \varphi(n_j)$ in the document graph corresponding to edges n_i, n_j in the query graph. When T is the arithmetic average, we get

$$\frac{1}{\mid E \mid} \sum_{(n_i,n_j) \in E} w_{arc}(\varphi(n_i), \varphi(n_j)) \tag{7}$$

Function 6 plays the same role as the objective function in standard fuzzy graph matching algorithms [Gol96], expressing the degree of membership of a candidate subgraph in the result set as a conjunction of the weights on corresponding arcs. In the third step, we output the matching subgraph and its rank J. As a very simple example, when we execute the blind query of Fig.3 on the graph of Fig. **??**, the cardinality $\mid E \mid$ of the edge set of the match is 1, and the rank function value for this match (when T is the average) is simply $J = w_{arc}(dept, contact) = 0.7$ (looked up from Tab.3).

5 Conclusion

We have presented what we consider to be a novel fuzzy technique to execute blind XML queries. The aim of this technique is to increase flexibility; indeed, while throughout the paper we supposed query semantics to be chosen *a priori*, employing the arithmetic mean as a compensative conjunction, a t-norm could also be chosen *ex-post* by presenting to the user a few prototypical examples. Though we are aware that much work is still to be done to tune our approach and validate it through experimentation, we believe some important points were addressed in this paper, while others were highlighted for future research.

References

[Bos98] P. Bosc On the Primitivity of the Division of Fuzzy Relations *Soft Computing*,vol. 2 n. 2, 1998

[Bue81] D.A. Buell A General Model of Query Processing in Information Retrieval Systems Information Processing & Management, vol.17 n.5, 1981 *Soft Computing*,vol. 2 n. 2, 1998

[Cer99a] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, L. Tanca XML-GL: A Graphical Language for Querying and Restructuring XML Documents *Computer Networks*, vol. 31, 1999

[Cla95] C.L. A. Clarke, G.V. Cormack, F. J. Burkowski An Algebra for Structured Text Search and a Framework for Its Implementation *The Computer Journal*, vol. 38 n.1, 1995

[Fag96] R. Fagin Combining Fuzzy Information from Multiple Systems, *Proc. of the Fifteenth ACM Symposium on Principles of Database Systems* Montreal, Canada, 1996

[Cer99b] S. Ceri, A. Bonifati Comparison of XML Query Languages SIGMOD Bullettin, to appear

[Cha92] K.P. Chan, Y.S. Cheung Fuzzy Attribute Graph with Applications to Character Recognition *IEEE Trans. on Systems, Man and Cybernetics* vol.22 no. 1, 1992

[Coh93] R. Cohen, G. Di Battista, A. Kanevsky, R. Tamassia Reinventing the Wheel: An Optimal Data Structure for Connectivity Queries *Proc. of ACM-TOC Symp. on the Theory of Computing*, S.Diego, CA, US, 1993

[Com98] S. Comai, E. Damiani, R. Posenato, L. Tanca A Schema-Based Approach to Modeling and Querying WWW Data In H. Cristiansen, ed., *Proceedings of Flexible Query Answering Systems, FQAS '98*, Roskilde (Denmark) *Lecture Notes in Artificial Intelligence* 1495, Springer, 1998.

[Clu98] S. Cluet, C. Delobel,J. Simeon, K. Smaga Your Mediators Need Data Conversion *Proc. Of ACM-SIGMOD Intl. Conf. on Management of Data*, 1998

[Clu99] S. Cluet, A. Deutsch, D. Florescu, A. Levy, D. Maier, J. McHugh, J. Robie, D. Suciu, J. Widom XML Query Languages: Experiences and Exemplars `http://www-db.research.bell-labs.com/user/simeon/xquery.html`

[Deu99] A. Deutsch, M. Fernandez, D. Florescu, A. Levy and D. Suciu *A Query Language for XML* Proceedings of the WWW-8 Intl. Conference, Canada, 1999

[Del98] A. Del Bimbo, E. Vicario Using Weighted Spatial Relatioship in Retrieval By Visual Content *Proc. IEEE Workshop on Content Based Access of Images*, Santa Barbara, CA, US, 1998

[Dub88] D. Dubois,R. Martin Clouaire,H.Prade Practical Computing in Fuzzy Logic in M.M. Gupta, T. Yamakawa, eds., *Fuzzy Computing*, North Holland, 1988

[Dub96] D. Dubois, H. Fargier, H. Prade Refinements of the Maximum Approach to Decision Making in Fuzzy Environments *Fuzzy Sets and Systems*, vol.81, 1996

[Dub98] D. Dubois, F. Esteva, P. Garcia, L. Godo, R. Lopez de Mantaras, H. Prade Fuzzy Set Modelling in Case-Based Reasoning, *Int. Jour. of Intelligent Systems* vol. 13, 1998

[Dub99] D. Dubois,H. Prade, F. Sedes Fuzzy Logic Techniques in Multimedia Database Querying: A Preliminary Investigations of the Potentials in R. Meersman, Z. Tari and S. Stevens, eds., *Database Semantics: Semantic Issues in Multimedia Systems*, Kluwer Academic Publisher, 1999

[Gol96] S. Gold, A. Rangarajan A Graduated Assignment Algorithm for Graph Matching *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, 1996

[Gup91] M.M. Gupta, J. Oi Theory of T-Norms and Fuzzy Inference Methods *Fuzzy Sets and Systems*, vol.40 n.3 1991

[Mic99] Microsoft Corp. XML-Data Specification `msdn.microsoft.com/xml/articles/xmldata.html`

[Kli88] J. Klir Fuzzy Sets and Information J. Wiley, 1988 Fuzzy Sets

[QL98] The Query Language Workshop `http://www.w3.org/TandS/QL/QL98` `www.w3.org/xml/xql98`

[Rob99] J. Robie The Design of XQL `www.texcel.no/whitepapers/xql-design.html`

[XSL] World Wide Web Council XSL Transformations, Version 1.0 W3C Recommendation, `www.w3.org/TR/1999/REC-xslt-19991116`

[XML] World Wide Web Council Extensible Markup Language, Version 1.0 W3C Recommendation, `www.w3.org/TR/1998/REC-xml-19980210`

XML and Meta Data Based EDI for Small Enterprises

Wolfram Wöß

Institute for Applied Knowledge Processing (FAW)
Johannes Kepler University Linz, Austria
e-mail: wwoess@faw.uni-linz.ac.at

Abstract. Today in many cases electronic data interchange (EDI) is limited to large scale industry connected to their own value added networks. Small-scale enterprises are not yet integrated in the communication flow, because actual EDI solutions are to complex, to inflexible or to expensive.
The approach presented in this paper separates knowledge about data structures and data formats from the process of generation of destination files. This knowledge is transformed into a meta data structure represented by XML document type definitions (DTD) which itself are stored within a database system. If any changes of the data interchange specification are necessary, it is sufficient to update the corresponding meta data information within the XML DTDs. The implementation of the data interchange processor remains unchanged. This type of adaptation does not require a software specialist and therefore it meets an important requirement of small-scale enterprises. Data transmission is done using the advantages of XML and Internet technology.

Key words. electronic data interchange (EDI), meta data, extensible markup language (XML), small-scale enterprises.

1 Introduction

Electronic data interchange (EDI) is not a very new topic in computer science and computer technology. The basic idea of EDI is interchange of structured data (business documents, business data, customer information, ...) between applications of participating communication partners without human intervention based on electronic transport media [1], [5].

Today Internet technology offers a medium for *low-cost* EDI. The following statement takes the importance of Internet technology as basic medium for EDI into account: EDI is electronic interchange of business documents, business data and information using a public standard format. Increasingly the medium for sending and receiving EDI documents is the Internet [...]. In practice, EDI is an application-to-application system, where two EDI systems (applications) are communicating with each other without human intervention [2].

An important aspect of EDI is that the used data structures and protocols are standardized [3]. But in the last years many different standards have been developed, most of them suitable for a special business sector. Examples are: ODETTE (Organization for Data Exchange by Tele Transmission in Europe) specialized for automobile industry, VDAFS (Verband der Automobilindustrie Flächenschnittstelle)

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 357–365, 2000.

specialized for automobile industry, SEDAS (Standardregelungen einheitlicher Datenaustauschsysteme) specialized for trade companies, IGES (Initial Graphics Exchange Specification) and STEP (Standard for the Exchange of Product data) specialized for producing enterprises. Some of these standards are in addition characterized by national restrictions.

This scenario of a large number of *concurrent* EDI standards leaded to the specification of ANSI X12 (American National Standard Institute) [2] and UN/EDIFACT (United Nations / Electronic Data Interchange For Administration, Commerce and Transport). Both standards meet the demands for independence from manufacturer, hardware, software, national aspects and business area.

By exemplary discussing the EDIFACT standard, today in many cases the following EDI scenario can be found out [1], [4]:

- Several versions of the EDIFACT specification have been developed.
- Incompatible standard-versions of EDIFACT are used.
- The result are several subsets (sub-standards) of EDIFACT which are only suitable for special applications. This is in contrast to the original idea of a common standard.
- The specification of the EDIFACT standard is complicated and therefore introduction and application of EDIFACT is time and cost intensive.
- EDIFACT was defined for machine processing and therefore EDIFACT messages are difficult to read for humans.

The mentioned drawbacks can be categorized into four problem areas:

- *Standards:* There are complicate standards which are characterized by an inflexible standardization procedure and which are suitable only for a few application scenarios.
- *Message-orientation:* Classic EDI focuses only messages without consideration of the corresponding business processes and interactions.
- *Interpretation:* There are complex interpretation processes because EDI messages are defined for machine processing.
- *Costs:* The introduction of EDI is very time and cost intensive (hardware, software and network resources).

Due to the discussed problems, today EDI is still limited to large-scale industry characterized by large and powerful EDP departments which are connected to VANs (value added networks). VANs are networks which are establish by third party companies. All administration tasks of a VAN are served by the VAN provider.

Small-scale enterprises are not yet integrated in these communication possibilities, because actual EDI solutions are to complex, to inflexible or to expensive. The special requirements of small-scale enterprises are

- high flexibility,
- lean administration and
- low costs.

The approach presented in this paper focuses the special requirements of small-scale enterprises. A configurable interface for EDI based on XML (eXtensible Markup Language) [6], [7] and meta data is introduced, which allows to administrate several

interfaces for data interchange with other business partners. The basic concept of this approach is to transform information about data interchange formats and data structures in the form of meta data represented by XML document type definitions (DTDs). Data interchange is possible in both directions, to and from a communication partner.

The paper proceeds as follows: Section 2 starts with a discussion of general problems with EDI in small-scale enterprises, which are characterized by lean organizations and low EDP budgets. In Section 3 the concept of the meta data based data interchange processor is introduced. This section includes a detailed description of the XML based architecture and the mapping mechanisms between business objects and source/destination files. Section 4 concludes and gives an outlook on future research.

2. Problems with EDI in Small-scale Enterprises

In general, there are two contrary ways to solve the data interchange problem:

- Implementation of one data interface per EDI communication partner. In the worst case *n* EDI partners require *n* different interfaces (Figure 1).
- Definition of a *common standard* which covers nearly all possible data interchange scenarios. This approach results in a complex interface specification with a more or less large overhead.

In the first case the implementation and maintenance of the whole set of interfaces is very time and cost intensive.

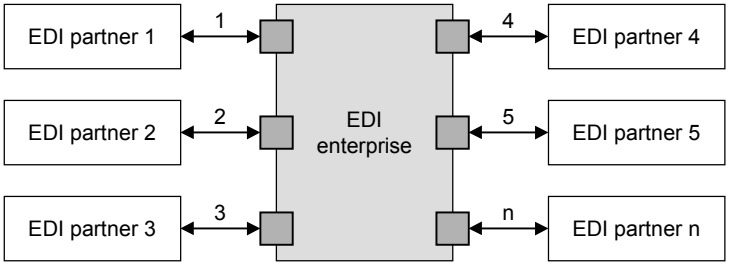


Fig. 1. Increasing number of data interfaces.

EDIFACT is a representative example for the second case, because it is a common standard which dominates wide areas of business to business EDI. The basic components of EDIFACT are a vocabulary (specified in directories), character sets and its grammar. EDIFACT is based on pre-defined separation signs which separate single data elements from each other.

The vocabulary of EDIFACT consists of data elements, data element groups, segments, messages and message groups. These components are organized in the form of a hierarchical order. Figure 2 shows this hierarchy up to data element groups.

The set of segments which is necessary to describe a business transaction is called message. Single messages are itself combined to message groups. In general an EDIFACT communication consists of a set of messages groups which are integrated to a single EDIFACT file. The communication between two EDIFACT partners is asynchronous.

The components of an EDIFACT file are specified in "directories", which are updated periodically to fulfill industrial demands. A correct EDIFACT communication requires identical versions of EDIFACT components at both communication partners [1]. This demand significantly increases the administration effort which is necessary to establish a correct and efficient EDIFACT communication.

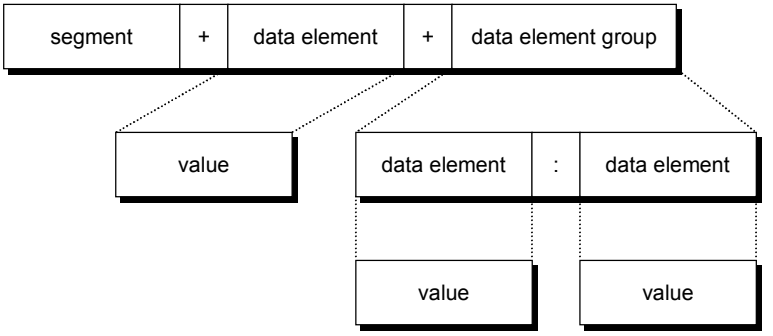


Fig. 2. Structure of an EDIFACT segment.

In general, large-scale enterprises have powerful organizations including an EDP department. Though the application of EDIFACT results in additional administration tasks, enterprises of this type make use of the rationalization potential in this field. Today in many cases the following scenario can be found out: a large-scale enterprise dominates a group of smaller suppliers by giving detailed specifications of the communication operations, including the physical network, the communication protocol and the data structures. In this case the advantages of EDI are only for the benefit of the large enterprise [5].

To overcome some of the discussed problems and to focus the requirements of small-scale enterprises, EDI clearing centers have been established. The main functionality of an EDI clearing center is the conversion of plain data into the requested EDIFACT data format in both conversion directions. The consequence is that the administration effort to establish EDIFACT data interchange is transferred from a small enterprise to the EDI clearing center. But at the same time a third communication partner, the EDI clearing center, is involved with the data interchange process. On the one hand this approach decreases the administration costs of the smaller enterprise on the other hand the EDI process is more inflexible. As a second aspect the EDIFACT overhead has to be considered. Especially for small-scale enterprises the question is: Why put up with the whole EDIFACT overhead if only a very simple data structure has to be transferred?

3. The Meta Data Based Data Interchange Processor

In contrast to many existing systems the approach presented in this paper separates knowledge about data structures and data formats from the process of generation of destination files and electronic transmission. This knowledge is transformed into a meta data structure represented by XML DTDs which itself are stored within a relational database system. After a request, the data interchange processor transforms the required information into the destination format which depends on the corresponding meta data. If any changes of the data interchange specification are necessary, it is sufficient to update the corresponding meta data information within the XML DTDs. The implementation of the data interchange processor remains unchanged. This type of adaptation requires neither a database administrator nor a software engineer and therefore the introduced concept meets the special requirements of small-scale enterprises.

To establish such a system three components are necessary (Figure 3):

- *Meta data* about data structures and data formats, represented by XML DTDs which are stored within a central database.
- An *interface management* module is necessary for the administration of meta data information which represents an individual data interface to an EDI communication partner.
- The *data interchange processor* in the first step receives input data which has to be transformed into the required destination format. In the next step the processor scans the meta data information corresponding to the required destination format. Based on this meta data the destination data file is generated dynamically.

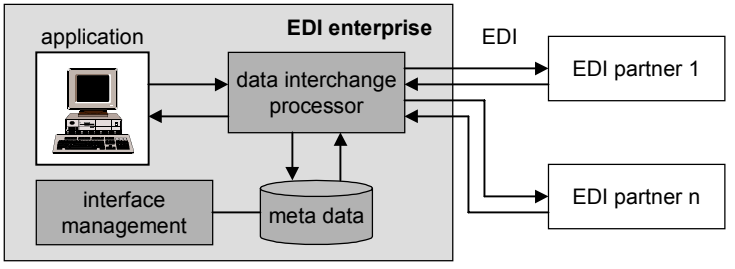


Fig. 3. Components of meta data based EDI.

The first prototype of the data interchange processor was based on a meta data structure which was mainly managed using two tables (EDI_partner_info, destination_file_field_order) of a relational database system. The destination file format was ASCII (American Standard Code for Information Interchange). In this case the EDI partner has to process the ASCII-file as input for its own software system(s). Since the data interchange processor does not determine the type of electronic transmission of the destination files, especially for small-scale enterprises using Internet technology offers many additional advantages. Beside low costs the advantages of an Internet solution for electronic transmission of data files are its availability, flexibility and the standard protocol TCP/IP (Transmission Control

Protocol/Internet Protocol). Moreover, a number of services like WWW (World Wide Web) and e-mail are available.

At this point the consideration is, to use Internet technology not only for electronic transmission but also for meta data specification and for generation of destination files, with the consequence that Internet technology is fully integrated as uniform platform in the whole EDI process.

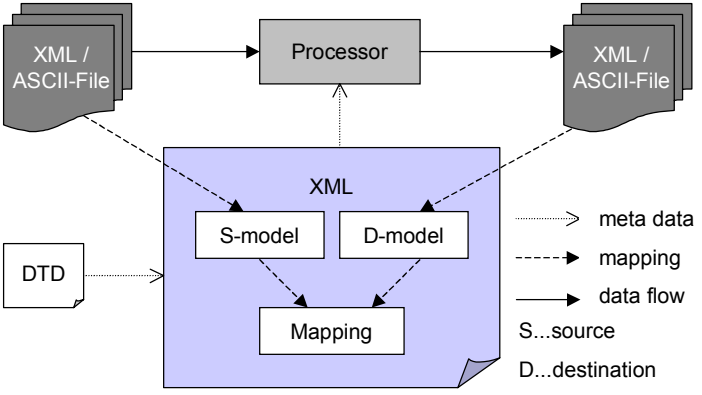


Fig. 4. XML based architecture.

Beside several other key characteristics XML is also a new and flexible concept for the specification of an EDI message. In contrast to HTML (HyperText Markup Language) which aims at the *presentation* of information the new meta language XML focuses on *structuring* of information which is also the most important process of an EDI application. Hence, for example, it is possible to model existing message types of EDIFACT as XML messages. Beside these aspects a further advantage of XML is that a simple Web-browser is able to process XML documents. If the destination files of an EDI process are transformed into a XML format, the EDI partner does not require a special software to process the destination files because a Web-browser is sufficient.

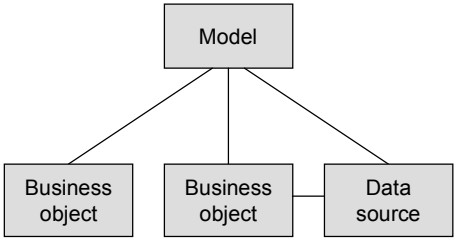


Fig. 5. Source and destination data models.

Taking these considerations into account the latest development of the introduced data interchange processor is based on XML DTDs (Figure 4). Within the XML based architecture source and destination files are in ASCII- (due to high compatibility) or

XML-format. Meta data information for the specification of the transformation of a data source into its destination format is represented by XML DTDs which are itself stored within a relational database system.

A DTD describes the mapping between the source model and the destination model each representing the data structures of the corresponding source/destination files. A model itself consists of business objects and the corresponding source or destination files (Figure 5). Business objects are representations of the data structures within the source or the destination files. A business object is designed as a tree structure and may itself consist of business objects and/or simple attributes (Figure 6). In practice this also allows the representation of recursive data structures which, for example, are necessary for the implementation of bill of material trees. Each attribute of a source model knows its corresponding data field within the source file and each attribute of a destination model references the corresponding destination file and data type.

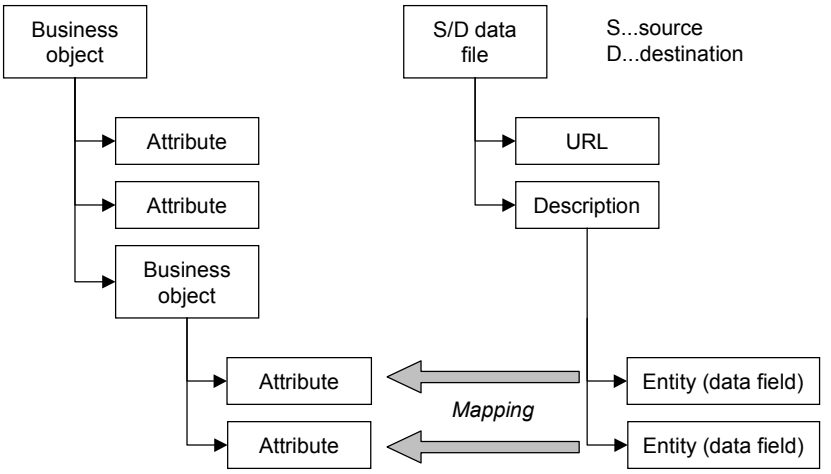


Fig. 6. Business object mapping.

The meta data information represented by XML DTDs is stored within a central database system to provide easy access for each software system within an enterprise. The maintenance of the meta data information is done by using an interface management module which allows to specify or update DTDs. Due to the central database system changes of the transformation specification (meta data) are immediately valid for all interacting software system.

A further key aspect of the XML based architecture is the possibility to specify layout information of an XML document separated from the XML document itself. Hence, it is possible to generate different outputs of one XML document using different layout styles which are defined with XSL (XML Stylesheet Language) [6]. In this case it does not take great effort to provide different outputs of the same EDI process for different enterprises.

The main advantages of XML based EDI are high flexibility and low implementation costs. In combination with XSL XML is important for structuring

information *and* for its presentation. EDI messages are not longer specialized for machine-processing but also easy to ready for humans.

A prototype of the meta data and XML based data interchange processor is already finished. In the next step we will focus on

- the integration of customers and
- the cooperation between the departments of an enterprise.

Both aspects require improved and easy to use EDI features. The WWW provides a unified platform in combination with an easy-to-use user interface. The implementation of a customer order form on a WWW client demonstrates the wide spectrum of EDI possibilities. It reaches from a simple transmission of data in one direction from a customer to an enterprise up to bi-directional transmission of data based on Java technology [8]. In this case the Web-browser is not only a own-way data presentation interface but a real two-way application. In addition the introduction of VRML (Virtual Reality Modeling Language) makes the presentation of 3-dimensional data (pictures) on a WWW client possible [9]. VRML is not a programming language but a modeling language which allows to model 3D objects. The main advantage of VRML is that the visualization of data is independent from the generating system (for example a CAD system). But VRML is not only useful for visualization of data. It also allows to define behavior of objects and therefore VRML makes process simulations possible [8]. VRML is a powerful extension of conventional WWW applications, especially for EDI between several departments of an enterprise (CAD data) as well as between a customer and an enterprise. For example, the customer specification of a final product may be generated and immediately visualized on a WWW client.

4 Conclusions

As discussed in Section 1 and Section 2, in many cases EDI is still limited to large-scale industry characterized by large and powerful EDP departments which are connected to VANs. Small-scale enterprises are not yet integrated in these communication possibilities, because actual EDI solutions are to complex, to inflexible or to expensive.

The approach presented in this paper focuses the special requirements of small-scale enterprises: high flexibility, lean administration processes and low costs. A configurable interface for EDI based on XML and meta data is introduced, which allows to administrate several interfaces for data interchange with other business partners efficiently.

In contrast to many existing systems the presented approach separates knowledge about data structures and data formats from the process of generation of destination files and electronic transmission. This knowledge is transformed into a meta data structure represented by XML DTDs which itself are stored within a central database system. After a request, the data interchange processor transforms the required information into the destination format which depends on the corresponding meta data. The main advantage of this concept is that if changes of the data interchange specification to or from an EDI partner are necessary, it is sufficient to update the

corresponding meta data information within the XML DTDs. The implementation of the data interchange processor remains unchanged. This type of adaptation requires neither a database administrator nor a software engineer and therefore this approach meets the special requirements of small-scale enterprises. Data transmission is done by using Internet technology.

Further work will be concentrated on additional integration of WWW and Internet technology. Focus of interest will be EDI between customers and an enterprise as well as cooperation between the departments of an enterprise.
The advantages of EDI based on Internet and WWW are [10]:

- low costs in comparison to many VANs,
- availability of several kinds of services based on the same platform and user interface,
- a uniform and standardized protocol (TCP/IP),
- ease-to-use services,
- multi-media presentation of data and
- linked information.

Additionally, new concepts and technologies like VRML open new possibilities for the implementation of Web based EDI.

References

1. Schmoll, Thomas: Handelsverkehr elektronisch, weltweit: Nachrichtenaustausch mit EDI/EDIFACT. Markt & Technik Verlag, München (1994)
2. Netscape: Electronic Data Interchange (EDI) Fundamentals. Netscape Communications Inc., California, USA (1998)
3. Picot, Arnold: Strategische, organisatorische und wirtschaftliche Perspektiven von EDI. In: Software AG (ed.): EDI Congress, Wiesbaden, BRD (1991)
4. Kalakota, R., Whinston, A.: Frontiers of Electronic Commerce. Addison-Wesley Publishing, USA (1996)
5. Alpar, P., Pickerodt, S.: Electronic Commerce im Internet – ein Überblick. Industrie Management 14 (1998), No. 1, GITO-Verlag Berlin (1998)
6. Behme, H., Mintert, S.: XML in der Praxis. Professionelles Web-Publishing mit der Extensiblen Markup Language. Addison-Wesley Publishing, BRD (1998)
7. Bradley, N.: The XML companion. Addison-Wesley Publishing, Great Britain (1998)
8. Knechtel, Ulrich: Produktdatenmanagement im Intranet/Extranet. Industrie Management 14 (1998), No. 1, GITO-Verlag Berlin (1998)
9. Hartmann, J., Wernecke, J.: The VRML 2.0 Handbook. Addison-Wesley Publishing USA (1996)
10. Lackes, Richard: Intranets als Teil betrieblicher Informationssysteme. Industrie Management 14 (1998), No. 1, GITO-Verlag Berlin (1998)

Optimal Page Ordering for Region Queries in Static Spatial Databases

Dae-Soo Cho, Bong-Hee Hong

Department of Computer Engineering, Pusan National University
Chang-Jun-Dong, Kum-Jung-Gu, Pusan, South Korea
{dsjo, bhhong}@hyowon.cc.pusan.ac.kr

Abstract. Page ordering is to define the order of pages in one-dimensional storage for storing two-dimensional spatial data to reduce the number of disk seeks. Previous works relating to page ordering have only used the space filling curves, especially the Hilbert curves. Page ordering based on the space filling curves does not take into account the uneven distribution of spatial data and the types of spatial queries. In this paper, we develop a cost model to be used for defining the problem of page ordering based on performance measurement and then find out the optimal page ordering based on the cost model for processing region queries in static databases. The experimental results show that the newly proposed ordering method can achieve considerable improvement over the previous ordering methods.

1 Introduction

A number of algorithms for clustering spatial data to reduce the number of disk seeks required to process spatial queries have been developed. One of the algorithms is the scheme of page ordering, which is concerned with the order of pages in one-dimensional storage for storing two-dimensional spatial data. Figure 1 shows the number of disk seeks can vary according to the order of a set of pages. Little work has been done to impose a linear ordering on pages. The goal of page ordering is that all of the pages, which are near in the data space, should be stored adjacently in a linear order.

Fig. 1. Examples of different page ordering: If two pages {P2, P3} should be accessed for processing a region query, one seek is needed in the case of (a), whereas two seeks are needed in (b).

In the previous works [5, 6, 7, 8], the space filling curves were only used to impose an order on all of the pages and it was argued that the order based on Hilbert values was the best. The Hilbert ordering has been proven to achieve better performance in uniformly distributed spatial data owing mainly to the following benefits: First, two

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 366–375, 2000.

consecutive pages on a Hilbert curve are spatially adjacent to each other. In other space filling curves, such as curves based on z-ordering or on gray-code, there may exist two consecutive pages which are not spatially adjacent to each other. Second, each page has two spatially adjacent neighbors which are not far and away. Whereas in row-prime curve, some pages may have two spatially adjacent neighbors which are farther than that.

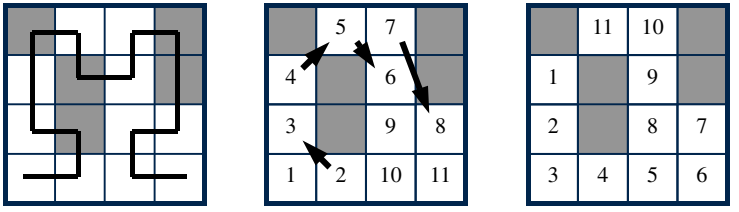


Fig. 2. The problem of page ordering based on a Hilbert curve: In the data space shown in (a), each cell corresponds to a page. The shaded cells represent the empty spaces. A sequence of numbers in the cells denotes the order of corresponding pages. In this case, the Hilbert order is shown in (b) and there are two consecutive pages which are not spatially adjacent to each other because the empty spaces are not considered. An example of an optimal order of pages in which no two consecutive pages are not spatially adjacent to each other is shown in (c).

The main motivation for this work is to uncover the drawback of the previous ordering method based on Hilbert values. With non-uniformly distributed spatial data, the most significant limitation of the Hilbert ordering is shown in Figure 2. To overcome this limitation of the Hilbert ordering method, we propose a new page ordering method in which the distribution of spatial data as well as the types of spatial queries are considered. To determine the optimality of a certain page order, we develop a new cost model of page ordering as a formula for performance measurement.

The next section reviews the related work for clustering in spatial databases. In Section 3, we propose the cost model of page ordering. In Section 4, we define the page ordering problem based on the cost model. In Section 5, we shortly discuss simple experimental evaluations. Finally, we present our conclusions and future works in Section 6.

2 Related Work

There are three kinds of works that are relevant to the study of reducing the cost of disk accesses in spatial databases: (1) single-page clustering (also called *packing techniques* in [9, 10], *optimal clustering* in [3, 12], and *local clustering* in [1, 2]), (2) multi-page clustering (also called *multi-page storage clustering* in [4] and *global clustering* in [1, 2]), and (3) the ordering of spatial objects or pages (also called *distance preserve mapping* in [5], *linear clustering of objects* in [7], and *optimal clustering of records* in [8], and *globally order preserving* in [6]).

In the single-page clustering, a set of spatial objects are stored in a page in order to speed up spatial query processing. Grouping 'spatially adjacent' objects into the same page leads to reduction in the number of page accesses. Most of the works on Spatial Access Methods (SAMs) fall under this category.

In contrast to the single-page clustering, the multi-page clustering stores a set of spatially adjacent objects in a set of physically consecutive pages (not in a page) which can be fetched by a single read request. The main idea of this method is to read both requested and non-requested pages together at one disk seek. In this method, additional data transfers, however, are expected due to the reading of non-requested pages. In [2], this overhead was alleviated by deciding whether a unit of disk access is a page or a cluster.

The page ordering in the previous works was to form a sequence of pages (or objects) determined by the order according to specific space filling curves. If two pages which are simultaneously requested are stored in sequence, the number of disk seeks can be reduced. This paper differs from the related works in the cost model to find the optimal page order. In the previous works, the ordering is based on the space filling curves. The limitation of traditional ordering methods is that the order is fixed regardless of the distribution of spatial objects.

3 Cost Model for Page Ordering

Let $DS = [0,1) \times [0,1)$ be the data space of unit square in which all the geometric objects are defined. Let $R = \{r_i \mid r_i$ is a sub-space of DS of rectangular shape$\}$ be a set of regions. The function *Area_of* can be defined as a generalized area function to calculate the area of an arbitrarily shaped sub-space. The region-area function $f_a : R \rightarrow RN$ (RN is the set of real numbers) is a specialized area function such that $f_a(r_i) = Area\_of(r_i)$ for $r_i \in R$. Let $Q = \{q \mid q \in R$ and q is a region of constant size with q_x as its width and q_y as its height$\}$ be a set of region queries. In this paper, we assume a query region to be a rectangular shape which has the same width and height. We also assume that all of the centers of query regions are uniformly distributed.

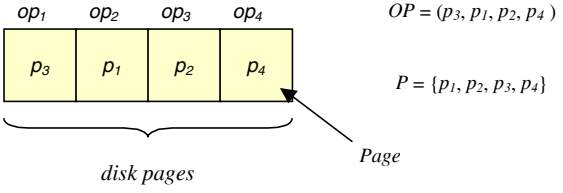


Fig. 3. For a page set $P = \{p_1, p_2, p_3, p_4\}$, the sequential order of pages is denoted by $OP = (op_1, op_2, op_3, op_4)$, where $op_1 = p_3$, $op_2 = p_1$, $op_3 = p_2$, and $op_4 = p_4$.

Let $P = \{p_1, p_2, .., p_k\}$ be a page set, and $OP = (op_1, op_2, .., op_k)$, $op_k \in P$ be an ordered page set of P. The ordered set is represented by using parentheses rather than braces. All of the pages are stored sequentially on disk according to the order listed in OP. For example, op_i and op_{i+1} are sequentially stored on disk, for $op_i, op_{i+1} \in OP$, $1 \le i \le k-1$. The page-region function $f_{ri}: P \rightarrow R$ is a function such that $f_{ri}(p_i) = r_i$, where r_i is

a minimal region enclosing all of the spatial objects in p_i. Figure 3 depicts the relationship between P and OP.

Let $I(r_i)$ be an inflated region of r_i by $q_x/2$ and $q_y/2$. $I(r_i)$ is a clipped region against DS. The domain of $I(r_i)$ is the set of center points of all region queries which intersect with r_i. $I(r_i)$ is characterized as follows.

$$Area\_of\ (\ I(r_i)\ ) = probability(\ query\ q\ retrieves\ page\ p_i\ ) \quad (1)$$

Eq.1 proposed in [3, 9] formulates the probability that the execution of a query q makes access to a page p_i. It means that a region query of which the center point is inside $I(r_i)$ retrieves a page p_i. [3, 9] also proposed the function of performance measurement $PM(P, Q)$ based on Eq.1. For a page set $P = \{p_1, p_2, .., p_k\}$ and a query set Q, the expected number of page accesses needed to perform a query $q \in Q$ is defined by PM, and is given by

$$PM(P,Q) = \sum_{i=1}^{k} probability(query\ q\ retrieves\ page\ p_i) = \sum_{i=1}^{k} f_a(I(f_{r_i}(p_i))) \quad (2)$$

The purpose of [3, 9] is to generate a page set for a given data set to reduce the cost of page accesses, whereas that of this paper is to generate an ordered page set for a given page set to reduce the number of seeks. We propose the function of performance measurement $PM(OP, Q)$ based on Eq. 2. The number of seeks (non-sequential disk accesses) is used to compute the performance measurement of query cost. $PM(OP, Q)$ is defined as the number of seeks, rather than the number of page accesses, and is given by

$$PM(OP,Q) = \sum_{i=1}^{k} probability(query\ q\ retrieves\ page\ op_i) - \sum_{i=1}^{k-1} probability(query\ q\ retrieves\ both\ pages\ of\ op_i\ and\ op_{i+1}) \quad (3)$$

Lemma 1 *For an ordered page set OP and a query set Q, the expected number of disk seeks, PM(OP, Q) is given by*

$$PM(OP,Q) = \sum_{i=1}^{k} f_a(I(r_i)) - \sum_{i=1}^{k-1} f_a(I(r_i) \cap I(r_{i+1})) \quad (4)$$

Proof: Let $N(k)$ be the expected number of seeks for the ordered page set $OP = (op_1, op_2, .., op_k)$ and the page set Q. We show that

$$N(k) = \sum_{i=1}^{k} f_a(I(r_i)) - \sum_{i=1}^{k-1} f_a(I(r_i) \cap I(r_{i+1})) \quad (5)$$

Induction base: For k=1,

$$N(1) = The\ expected\ number\ of\ page\ accesses = f_a(I(r_1)) \quad (6)$$

Eq. 6 is true, because the number of seeks is equal to the number of page accesses (Eq. 2) if there is only one page.

***Induction hypothesis*:** Assume, for an arbitrary positive integer k, that

$$N(k) = \sum_{i=1}^{k} f_a(I(r_i)) - \sum_{i=1}^{k-1} f_a(I(r_i) \cap I(r_{i+1})) \quad (7)$$

***Induction step*:** We need to show that

$$N(k+1) = \sum_{i=1}^{k+1} f_a(I(r_i)) - \sum_{i=1}^{k} f_a(I(r_i) \cap I(r_{i+1})) \quad (8)$$

For $OP = (op_1, op_2, .., op_k, op_{k+1})$, $N(k+1)$ is the sum of $N(k)$ and the additional expected number of seeks by op_{k+1}. $I(r_{k+1})$ of r_{k+1} derived from $f_r(op_{k+1})$ can be divided into two parts.

Part I: $I(r_k) \cap I(r_{k+1})$ that is the intersected region of $I(r_{k+1})$ and $I(r_k)$.

If the center point of q is inside $I(r_k) \cap I(r_{k+1})$, two pages, op_k and op_{k+1}, at least should be accessed. In this case, no more additional disk seeks in $N(k+1)$ related to $N(k)$ are required because op_k and op_{k+1} are stored sequentially on disk.

Part II: $(I(r_{k+1})\text{-}I(r_k))$ that is the rest space of $I(r_{k+1})$.

If the center point is inside $(I(r_{k+1})\text{-}I(r_k))$, the access of op_k is not required any more. This means that reading op_{k+1} requires one additional disk seek.

Therefore, $N(k+1)$ is the sum of $N(k)$ and the probability that processing q requires an access to op_{k+1} and does not require any access to op_k.

To show the Eq. 8,

$$\begin{aligned} N(k+1) &= N(k) + Area\_of(I(r_{k+1}) - I(r_k)) \quad (9) \\ &= \sum_{i=1}^{k} f_a(I(r_i)) - \sum_{i=1}^{k-1} f_a(I(r_i) \cap I(r_{i+1})) + f_a(I(r_{k+1})) - f_a(I(r_k) \cap I(r_{k+1})) \\ &= \sum_{i=1}^{k+1} f_a(I(r_i)) - \sum_{i=1}^{k} f_a(I(r_i) \cap I(r_{i+1})) \end{aligned}$$

We then proceed as before to conclude that Eq. 5 is true. ■

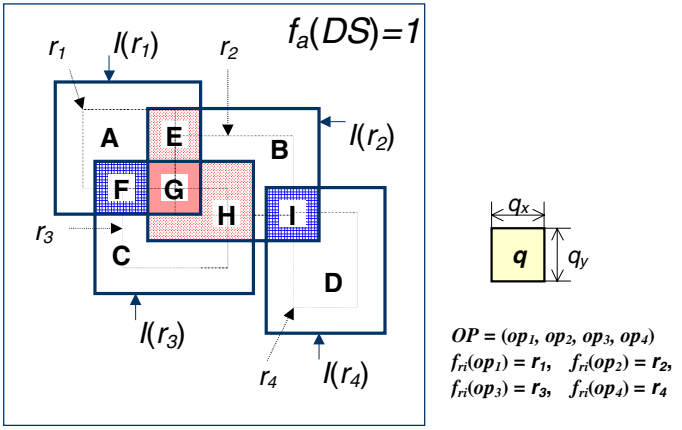


Fig. 4. The ordered page set and the inflated regions for example 1

Example 1. We show that Eq. 4 represents the expected number of seeks for an ordered page set OP and a query set Q. Figure 4 shows the inflated regions of $f_{ri}(op_i)$, $op_i \in OP$ inflated by $q_x/2$ and $q_y/2$. If the center point of a query is inside a sub-space marked with a capital letter, then;

1. A, B, C, and D : one page access and one disk seek are required
2. E and H : two page accesses and one disk seek are required
3. F and I : two page accesses and two disk seeks are required
4. G : three page accesses and one disk seek are required

Therefore, the average number of seeks for each query $q \in Q$ is calculated by Eq. 10.

$$\begin{aligned} \textit{The number of seeks} &= 1 * \textit{probability}(1\ \textit{seek}) + 2 * \textit{probability}(2\ \textit{seeks}) \\ &= 1 * \ \textit{Area\_of}(A+B+C+D+E+G+H) + 2 * \textit{Area\_of}(F+I) \end{aligned} \quad (\mathbf{10})$$

According to Lemma 1, the number of seeks can also be calculated by Eq. 11. The results of both equations are the same.

$$\begin{aligned} PM(OP,Q) &= \sum_{i=1}^{4} f_a(I(r_i)) - \sum_{i=1}^{3} f_a(I(r_i) \cap I(r_{i+1})) \\ &= \textit{Area\_of}(A+E+F+G) + \textit{Area\_of}(B+E+G+H) \\ &\quad + \textit{Area\_of}(C+F+G+H) + \textit{Area\_of}(D+I) \\ &\quad - \textit{Area\_of}(E+G) - \textit{Area\_of}(G+H) \\ &= 1 * \ \textit{Area\_of}(A+B+C+D+E+G+H) + 2 * \textit{Area\_of}(F+I) \end{aligned} \quad (\mathbf{11})$$

4 Page Ordering Problem

In this section, we define the page ordering problem based on $PM(OP, Q)$ proposed in the previous section and show that the problem is equivalent to the traveling salesman problem.

Definition 1. Page Ordering Problem(POP)
Given a page set P and a query set Q, *POP* is to determine an ordered page set OP for which $PM(OP, Q)$ is minimal.

$PM(OP, Q)$ consists of two sub-functions: $\Sigma f_a(I(r_i))$ (independent of page order) and $\Sigma f_a(I(r_i) \cap I(r_j))$ (dependent on page order). Determining an ordered page set OP for which $PM(OP, Q)$ is minimal, is equivalent to determining an ordered page set OP for which $\Sigma f_a(I(r_i) \cap I(r_j))$ is maximal. We conclude that *POP* is very similar to the well-known minimum weighted Hamiltonian path problem, without specifying the starting point and the terminating point. For a given page set P, we can obtain a corresponding weighted graph $H = (V, E, w)$ as follows:

$$V = P,$$
$$E = \{(p_i, p_j) \mid p_i, p_j \in P, i \neq j\},$$

and,

$$w: E \rightarrow RN,\ w((p_i, p_j)) = 1\text{-}f_a(I(r_i) \cap I(r_j))$$

where RN is the set of real numbers. The weight of an edge is computed as the total area of a data space minus the area of $f_a(I(r_i) \cap I(r_j))$ to convert a maximization problem into a minimization problem.

It is known in [13] that a minimum weighted Hamiltonian path problem can be treated as a standard Traveling Salesman Problem (TSP).

Lemma 2 *POP* is equivalent to *TSP* for a weighted graph $G = (V, E, w)$ such that

$$V = P \cup \{v_0\},$$

where $P = \{p_1, p_2, .., p_k\}$ is a set of pages and v_0 is an artificial vertex to solve the minimum weighted Hamiltonian path problem by *TSP*,

$$E = \{(p_i, p_j) \mid p_i, p_j \in P, i \neq j\} \cup \{(p_i, v_0) \mid p_i \in P\},$$

and

$$w: E \rightarrow RN.$$

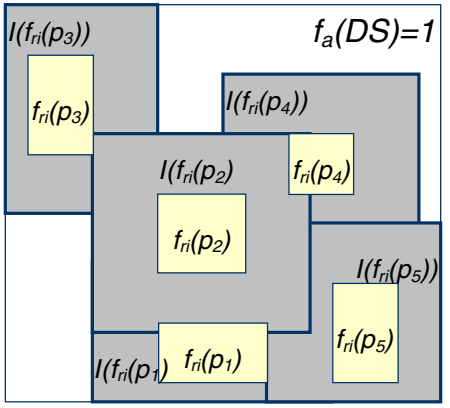
Edge weights are computed as follows:

1. For edge $e = (p_i, p_j) \in P^2$, $w((p_i, p_j)) = 1\text{-}f_a(I(r_i) \cap I(r_j))$.
2. For edge $e = (p_i, v_0) \in P \times \{v_0\}$, $w((p_i, v_0)) = 0$.

Proof: The Graph G is Hamiltonian, because G is a complete and weighted graph. Let $C = \{(v_0, op_1), (op_1, op_2), .., (op_{k-1}, op_k), (op_k, v_0)\}$, $op_i \in P$ be a minimum weighted Hamiltonian cycle in G. By removing two edges incident with the vertex v_0 from C, we can get a minimum weighted Hamiltonian path L in G from op_1 to op_k. Now we can easily get a solution of *POP* from a solution of the corresponding *TSP* by interpreting a minimum weighted Hamiltonian path L as an ordered page set $OP = (op_1, op_2, .., op_k)$. From the viewpoint of constructing a corresponding weighted graph, it is obvious that the solution of *POP* is identical with that of *TSP*. ■

Example 2. Figure 5 shows the graphical display of the inflated regions for a page set $P = \{p_1, p_2, p_3, p_4, p_5\}$ and a query set Q. The area of all of the queries in Q is 9% of the data space. To determine an optimal ordered page set OP, we have to construct a corresponding weighted graph according to Lemma 2. Figure 6 shows the weighted

graph for the page set and the query set. The lower the weight of an edge is, the higher the possibility of simultaneously requiring the corresponding pages of vertices at each end of the edge is. The least number of seeks can be obtained by processing a query in an ordered page set which has minimal weight.



$f_a(I(f_{ri}(p_1)) \cap I(f_{ri}(p_2))) = 0.087$
$f_a(I(f_{ri}(p_1)) \cap I(f_{ri}(p_5))) = 0.052$
$f_a(I(f_{ri}(p_2)) \cap I(f_{ri}(p_3))) = 0.030$
$f_a(I(f_{ri}(p_2)) \cap I(f_{ri}(p_4))) = 0.060$
$f_a(I(f_{ri}(p_2)) \cap I(f_{ri}(p_5))) = 0.017$
$f_a(I(f_{ri}(p_4)) \cap I(f_{ri}(p_5))) = 0.026$
$f_a(I(f_{ri}(p_1)) \cap I(f_{ri}(p_3))) = 0$
$f_a(I(f_{ri}(p_1)) \cap I(f_{ri}(p_4))) = 0$
$f_a(I(f_{ri}(p_3)) \cap I(f_{ri}(p_4))) = 0$
$f_a(I(f_{ri}(p_3)) \cap I(f_{ri}(p_5))) = 0$

Fig. 5. The inflated regions for example 2

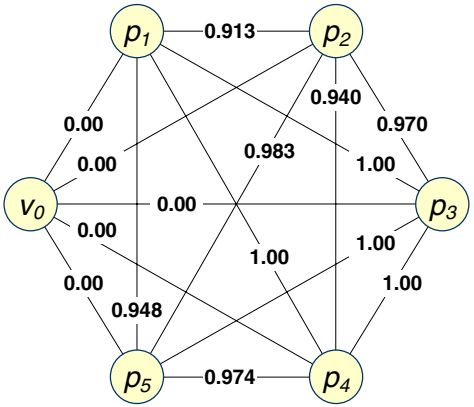


Fig. 6. A weighted graph G to solve a *POP* in example 2: An artificial vertex v_0 and edges from v_0 to all the vertices with weight 0 are inserted to solve a minimum weighted Hamiltonian path problem as a standard *TSP*.

In *TSP* with n cities, there are $(n\text{-}1)!/2$ possible Hamiltonian cycles. Therefore, the number of all possible Hamiltonian cycles is 60 in this graph and there exists the same number of Hamiltonian paths derived by removing the vertex v_0 in a Hamiltonian cycle. Each Hamiltonian path represents an ordered page set. Figure 7 shows three ordered page sets among the possible 60 ordered page sets for a given page set P. For a given page set P and a query set Q, in this example, the optimal ordering outperforms the Hilbert ordering by 6.5%.

notations	*ordered page sets*	Σw_i	*PM(OP, Q)*
OP_H	$(p_1, p_2, p_3, p_4, p_5)$ or $(p_5, p_4, p_3, p_2, p_1)$	3.857	0.865
OP_W	$(p_1, p_4, p_3, p_2, p_5)$ or $(p_5, p_2, p_3, p_4, p_1)$	3.953	0.961
OP_O	$(p_5, p_1, p_2, p_4, p_3)$ or $(p_3, p_4, p_2, p_1, p_5)$	2.853	0.809

Fig. 7. Several ordered page sets for example 2: : The order of pages in the first ordered page set denoted by OP_H is determined by the Hilbert value of a region $f_{ri}(p_i)$. OP_H is independent of the graph G. The Hilbert value of a center point of a region is used as the Hilbert value of the region. This ordering method has been used in the previous works[5, 7]. The second and the third ordered page sets are maximum and minimum weighted Hamiltonian paths in G, denoted by OP_W (the 'worst' case) and OP_O (the 'optimal' case), respectively. Recall that $PM(OP, Q)$ means the number of seeks for a region query $q \in Q$.

5 Experimental Evaluation

In this section, we investigate the performance of our page ordering method based on the cost model proposed in section 3, and compare it with the page ordering method based on Hilbert values. To solve the NP-hard problem of *POP*, we implement a heuristic based on the Simulated Annealing method which is one of the probabilistic hill climbing algorithms.

query size / data set	0.25	1.00	4.00	9.00	16.0
DS-A	17.62%	21.02%	22.78%	24.21%	24.95%
DS-B	14.81%	18.29%	20.25%	21.01%	22.75%

Fig. 8. Relative improvement of the optimal ordered page sets over $OP(H)$: The ratios of the occupied area of region queries are 0.25%, 1.00%, 4.00%, 9.00% and 16.0% of the data space. Experiments in DS-B shows less relative improvement than those in DS-A. This is because more states need to be explored in DS-B which consists of 3,490 pages which is a relatively larger number of pages compared to 1,022 in DS-A.

Our test data is based on polygonal data from the Sequoia 2000 benchmark. We used two test data sets: a data set DS-A that consisted of 1,022 pages with 17,048 spatial objects, and a data set DS-B that consisted of 3,490 pages with 65,157 spatial objects. All pages were stored on disk in sequence, and the order of each page was determined according to the page ordering method.

To evaluate the performance of the above mentioned page ordering methods, we did test for five kinds of query sets that are composed of region queries having different sizes. Figure 8 shows a relative improvement of optimal ordered page sets over $OP(H)$. The experimental results match exactly the expected results. The optimal ordered page set shows better performance over $OP(H)$.

6 Conclusion

In this paper, we proposed an optimal page ordering method to reduce the number of seeks in static spatial databases. To determine which ordered page set is optimal, we developed a cost model for page ordering. We defined the page ordering problem based on the cost model and showed that the problem is equivalent to the traveling salesman problem. Performance evaluation using the two data sets demonstrated the optimal page ordering performs from 15% to 25% better than the Hilbert ordering.

To investigate the performance of the page ordering method based on the cost model, the same size of query regions is used for building a query set. For generality, however, we need to use the different sizes of query regions. We will evaluate our optimal page ordering in a dynamic situation in order to make our ordering method to be more practical.

References

1. Thomas Brinkhoff, Holger Horn, Hans-Peter Kriegel, and Ralf Schneider, "A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems", pp357-376, SSD, 1993
2. Thomas Brinkhoff, and Hans-Peter Kriegel, "The Impact of Global Clustering on Spatial Database Systems", pp168-179, VLDB, 1994
3. Lukas Bachmann, Bernd-Uwe Pagel, and Hans-Werner Six, "Optimizing Spatial Data Structures For Static Data", pp247-258, IGIS, 1994
4. Cisbert Dröge, and Hans-Jörg Schek, "Query-Adaptive Data Space Partitioning using Variable-Size Storage Clusters", pp337-356, SSD, 1993
5. Christos Faloutsos, and Shari Roseman, "Fractanls for Secondary Key Retrieval", pp247-252, PODS, 1989
6. Andreas Hutflesz, Hans-Werner Siz, and Peter Widmayer, "Globally Order Preserving Multidimensional Linear Hashing", pp572-579, ICDE, 1988
7. H. V. Jagadish, "Linear Clustering of Objects with Multiple Attributes", pp332-342, SIGMOD, 1990
8. H. V. Jagadish, Laks V. S. Lakshmanan, and Divesh Srivastava, "Snakes and Sandwiches: Optimal Clustering Strategies for a Data Warehouse", pp37-48, SIGMOD, 1999
9. Ibrahim Kamel, and Christos Faloutsos, "On Packing R-trees, Information & Knowledge Management", pp490-499, CIKM, 1993,
10. Ibrahim Kamel, and Christos Faloutsos, "Hilbert R-tree: An improved R-tree using fractals", pp500-509, VLDB, 1994
11. Jack A. Orenstein, "Spatial Query Processing in an Object-Oriented Database System", pp326-336, SIGMOD, 1986
12. Bernd-Uwe Pagel, Hans-Werner Six, and Mario Winter, "Window Query-Optimal Clustering of Spatial Objects", pp86-94, PODS, 1995
13. G. Reinelt, "The Traveling Salesman Problem", Springer Verlag, 1994.
14. Bernhard Seeger, Per-Ake Larson, and Ron McFadyen, "Reading a Set of Disk Pages", pp592-603, VLDB, 1993
15. Gerhard Weikum, "Set-Oriented Disk Access to Large Complex Objects", pp426-433, ICDE, 1989

A Multi-Channel Dissemination System Based on Time-Series Clustering Mechanism for On-Line News Articles

Koichi MATSUMOTO † Kazutoshi SUMIYA ‡ Kuniaki UEHARA ‡

† Division of Computer and Systems Engineering,
Graduate School of Science and Technology, Kobe University
‡ Division of Urban Information Systems
Research Center for Urban Safety and Security, Kobe University
{koichi,sumiya,uehara}@ai.cs.kobe-u.ac.jp

Abstract. In this paper, we propose a multi-channel dissemination system with a clustering mechanism and a presentation technique for time-series on-line news articles on the Internet. We describe a detecting technique for articles whose topics are the same. These articles are called follow-up articles. In addition, we describe a calculating method for confidence level and scoop level assigned to news articles using temporal information. Furthermore we describe a prototype system called *Carthage* based on our proposed method. The system reconstructs all of the articles to follow-up article groups and distributes the article group with several user interfaces.

1 Introduction

In recent years, broadcast-based information dissemination systems[1] on the Internet are becoming popular due to rapid advances in the field of WWW technology and information dissemination, for example, PointCastNetwork[2] and other news dissemination systems. In conventional *pull-based* dissemination systems, information has to be retrieved and narrowed down by dialog operation. Therefor, it is thought that real time current-events news information is suitable for *push-based* dissemination. Nevertheless, there are some problems in conventional push-based dissemination systems:

- It is difficult to find target news articles because many news articles are disseminated and there are too many channels.
- There are no pointers to related articles on multi-channel dissemination systems, because news servers disseminate articles independently.

In this paper, we describe a multi-channel dissemination system for on-line news articles. The system detects follow-up articles supporting multi-channel based on the information of a time-series article. We propose a method of calculating a dissemination priority level based on time information. We describe the

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 376–385, 2000.

methods of synthesizing channels based on dissemination priority level and disseminating groups of time-series articles with high priority level. Furthermore, we describe the implementation issues of the prototype system called *Carthage*.

The rest of the paper is organized as follows: Section 2 describes the analysis of the properties of on-line news articles and follow-up articles. Section 3 explains our proposed mechanism for time-series articles. Section 4 describes a prototype system based on our proposed mechanism. Finally, Section 5 gives a conclusion.

2 On-Line News Article

2.1 Multi-channel dissemination system

Recently, there are on-line news channels such as "Yahoo News"[3], "Hot channel"[4] and "Lycos News"[5] as news dissemination servers on the Internet. These news channels disseminate real-time current-events news at regular intervals[1]. In some cases, a news article is disseminated and the correction of the news article is disseminated after a certain time. Also, additional information can be disseminated. It can be thought that there are several groups of news articles among all of the articles. That is, there are time-lined document whose topics are the same. We call these articles *follow-up articles*.

News servers disseminate time-series articles categorized into several channels based on their contents, such as *world* channel, *business* channel, *politics* channel, etc. In conventional systems, servers disseminate their articles independently and the clients recieve the articles directly. In our approach, all of the articles are filtered and constructed before the clients receive them (Fig. 1).

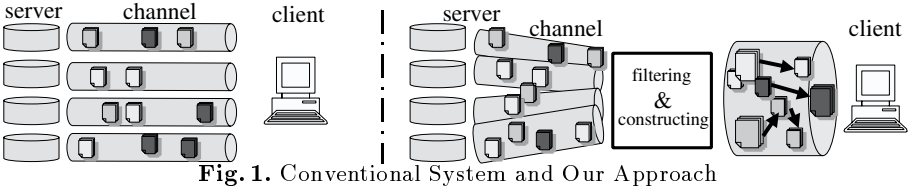


Fig. 1. Conventional System and Our Approach

2.2 Follow-up article

Since the contents of all follow-up articles are related to the contents of former time-series articles, we have to deal not with an independent follow-up article but with the earlier time-series article. We define articles related to the same topic as *follow-up article group* (hereafter referred to as FA).

Table 1. shows an example of an FA on CNN international news channel[6]. The table shows articles reported about Fijian coup d'etat occurred at May 19th. No. 1221 is the first report. No. 1345 is the additional information. No. 1670 is the continued situation.

[1] The dissemination from a server to clients is done at regular intervals as a pull-based manner. For users, however, it looks like push-based delivery.

No.	DATE	ARTICLE
1221	May,19 9:45a.m.	**Fiji PM and 7 Cabinet ministers held hostage** – Fiji's Prime Minister Mahendra Chaudhry and seven Cabinet ministers are reportedly being held hostage in the Fijian ...
	⋮	
1345	May,21 8:08a.m.	**Fijian rebels release hostages** – The armed rebels who occupied Fiji's parliament released 10 hostages on Sunday. Fiji radio announced Prime Minister Mahendra Chaudry will also be released ...
	⋮	
1670	June,7 6:37a.m.	**Fiji's military ruler stands firm against rebels** – Fiji's military government took a harder line Tuesday with armed rebels holding hostages in the nation's parliament ...

Table 1. Example of FA

2.3 Properties of the follow-up article groups

In this section, we describe the properties of FAs. We characterize three kinds of FA chaining patterns: (1)normal, (2)transit, and (3)return. It is important to note that transit and return patterns are observed among multi-channel dissemination.

Normal pattern In this pattern, all of the articles in a certain FA are in the same channel as shown in Fig. 2. For example, when a bank burglary incident happens, the first article would be disseminated immediately. In this case, follow-up articles such as the caught of the criminal may be disseminated after several days. The example of Table 1. is also of this pattern.

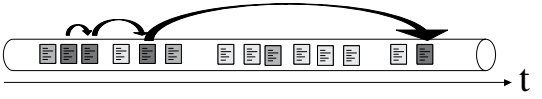


Fig. 2. FA on uni-channel

Transit pattern Fig. 3. shows this pattern. In channel B, there are three articles whose topics are the same. Then, in channel A, a new article whose topic is same as the three earlier articles is disseminated. The new article should be considered to be a follow-up article.

For example, news about the Taiwan Earthquake are disseminated frequently on the *international* channel. This situation continues, and after a while the news

which reports that memory modules for PCs got very expensive is disseminated on the *economy* channel. These events are very related, because most of memory modules are made in Taiwan whose economy was jolted and damaged by the earthquake.

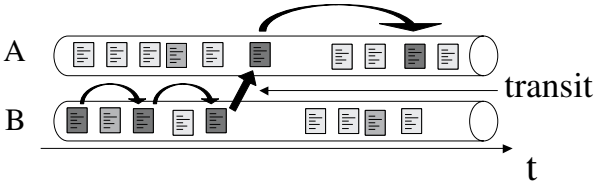


Fig. 3. Transit pattern

Return pattern Fig. 4 shows this pattern. In channel S and T, there was a transit pattern at an earlier time. Then, a transit from S to T may occur. This is called "return pattern", because the transit runs to the original channel. For example, the article which reports the recovery of the industry in Taiwan is disseminated after a while in the *international channel*.

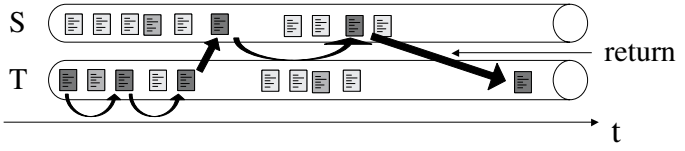


Fig. 4. Return pattern

3 Time-Series Clustering Mechanism

3.1 FA clustering mechanism

To detect a follow-up article, we use the feature vector of the article. When a new article a_{new} has been disseminated on a channel, the feature vector of the article is calculated. Then, the similarity between the feature vector and the feature vector of existing FAs is calculated by cosine similarity[2]. The cosine similarity of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is defined by the following formula:

[2] The feature vector of a FA is the mean vector of the feature vector of all articles contained in the FA.

$$sim(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|} \tag{1}$$

If the similarity exceeds a threshold and it is higher than its similarity with any other FA, the article is regarded as an element of the FA.

Furthermore, we propose a clustering mechanism for multi-channel dissemination. Several functions are used to add new articles to existing FAs. Basically, two functions measure certain aspects of classification as follows:

1. Similarity between the new article and an existing FA;
2. Time interval between the new article and the last article of an existing FA.

We can use the measurement of similarity described above for multi-channel dissemination. The measurement of interval for multi-channel environments is different from the measurement for uni-channel environments. If a new article is added to an existing FA in the same channel, the interval between the new article and the last article of the FA should not be limited. That is, no matter how late a new article arrives, it should be added to the existing FA. Of course, the similarity between the new article and the FA needs to be high.

On the other hand, if a new article belongs to another channel, the interval between the new article and the last article of the FA should be limited. If the interval is large, the article can not be a candidate for the FA. This is why the topic of the new article may be different from the FA according to characteristics of each channel. However, there are some cases when the new article should be added to the FA even if the interval is large. In this case, the FA should contain at least one article which belongs to the same channel of the new article. That is, the topic would go across several channels. This is the typical case of a return pattern.

Fig. 5 shows the limitation of the interval between the new article and the last article of the existing FA. The left figure shows the normal pattern, the middle one shows the transit pattern, and the right one shows the return pattern.

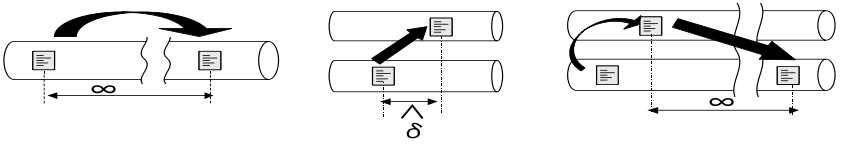


Fig. 5. Time interval of adding follow-up article

The clustering process for multi-channel dissemination is done through the following steps. The function $sim(\mathbf{x}, \mathbf{y})$ calculates cosine similarity between vector $\mathbf{x}$ and vector $\mathbf{y}$ and the function $interval(x, y)$ returns the dissemination time interval between article x and article y.

1. **foreach** $1 \leq i \leq n$
   if $(sim(FA_i, a_{new}) > sim(FA_{cand}, a_{new}))$
   $cand = i$;
   end
2. **else if** $sim(FA_{cand}, a_{new}) < \theta$
   then create $\langle FA_{new} \rangle$
3. **else if** $channel(a_{new}) = channel(last(FA_{cand}))$
   then $a_{new} \rightarrow \langle FA_{cand} \rangle$
4. **else if** $channel(a_{new}) \in channel(articles(FA_{cand}))$
   then $a_{new} \rightarrow \langle FA_{cand} \rangle$
5. **else if** $interval(last(FA_{cand}, a_{new})) < \delta$
   then $a_{new} \rightarrow \langle FA_{cand} \rangle$
6. create $\langle FA_{new} \rangle$

where $channel(a)$ returns which channel article a is disseminated, $articles(F)$ returns all of the articles contained in the FA, and $last(F)$ returns the last article of the FA.

Step 1. searches for the candidate from existing FAs for the new article. The FA whose similarity is highest becomes the candidate. **Step 2**. inspects whether the similarity runs over a threshold. If not, a new FA is created and the new article becomes the first article of the FA. **Step 3**. inspects if the last article of the candidate FA and the new article belongs to the same channel. If it is true, the new article connects to the FA as the last article (*normal pattern*). Otherwise, **Step 4**. checks whether the channel which the new article belongs to is contained in the channels which all of the article belongs to the candidate FA. If it is true, the new article connects to the FA as the last article (*transit pattern*). Otherwise, **Step 5**. inspects if the interval between the last article of the candidate FA and the new article is less than a threshold δ. If it is true, the new article connects to the FA as the last article (*return pattern*). Otherwise a new FA is created and the new article becomes the first article of the FA.

3.2 Dissemination priority of articles

In this section, we propose a priority for articles. Generally, more accurate and fresher articles are more valuable. We calculate the dissemination priority level of each article based on two independent factors: confidence level and scoop level.

Confidence level When the same information is disseminated from more than one source (channel), the information can be thought to be true, so confidence is higher. If there are articles which report about the same topic on other channels, the confidence of the later articles is higher. Then, as time passes, the confidence level gets higher generally.

For example, on TV news programs, if a headline such as *"A large-scale earthquake occurred in Kobe before sunrise today"* is reported on several channels,

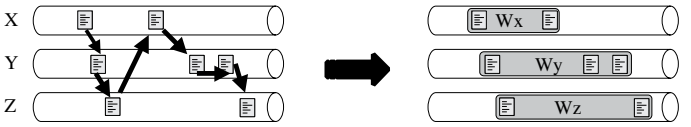


Fig. 6. Projected period of the FA on each channel

televiewers feel that the reported news is unambiguous. Although the first news and news reported near the time the event occurred may contain ambiguous information, news articles get accurate as time passes and the same topic news may be disseminated on other channels. In other words, in the case that only one channel reports the news, the confidence level is low, but as news on the same subject is disseminated on other channels and time passes, confidence level gets higher.

Therefore, the more the number of channels which disseminate articles about the same topic and the higher the frequency of the articles are disseminated, the higher the confidence level of the article is. We define the confidence level $Conf(a_n)$ of an article a_n as follows:

$$Conf(a_n) = \sum_{k=1}^{N} (W_{channel(a_n)} \odot W_k) \times delay(W_{channel(a_n)}, W_k) \tag{2}$$

where W_k is the projected period of the FA on the channel k. Fig. 6 shows the projected period: W_X, W_Y, W_Z. The operation $\odot$ means the overlap of the period of the FA. $X \odot Y$ shows the overlap of the periods X and Y. The function $delay(X, Y)$ returns the time lag between the periods X and Y, where X starts at time t_{s_X} and ends at time t_{e_X}, and Y starts at time t_{s_Y} and ends at time t_{e_Y}. If t_{s_Y} is greater than t_{s_X}, the function returns 0.

$$X \odot Y = |min(t_{e_X}, t_{e_Y}) - max(t_{s_X}, t_{s_Y})| \tag{3}$$

$$delay(X, Y) = \log((t_{s_Y} - t_{s_X}) + 1) \tag{4}$$

Fig. 7 shows the overlap and delay between the period which the article a_n belongs to and all of the periods on other channels.

Scoop level When an event happens, the first article is immediately disseminated in one channel earlier than other channels. If the document of the FA about a topic was disseminated a long time ago, a new time-series document about the same topic is disseminated. Let us consider the example of a bank burglary incident. The first article about the incident has a high scoop level. We define scoop level $Scoop(a_n)$ of an article as follows:

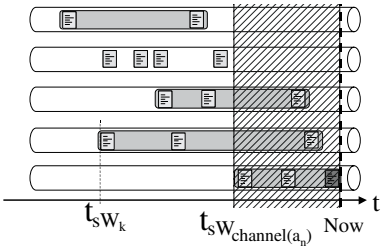


Fig. 7. Confidence level and Scoop level

$$Scoop(a_n) = \frac{N}{n\_channel(a_n)} \times \frac{1}{t_{a_n} - event(a_n)} \tag{5}$$

where $n\_channel(a_n)$ is the number of channels which contain the articles of the FA, N is the total number of channels, $event(a_n)$ is the time which the event of the article occurs actually, and t_{a_n} is the time when the article a_n is disseminated. Fig. 7 shows the time lag between t_{a_n} and the dissemination time of the first article of each channel.

The dissemination priority of an article is calculated based on the confidence level and scoop level as follows:

$$Priority(a_n) = w_1 \times Conf(a_n) + w_2 \times Scoop(a_n) \tag{6}$$

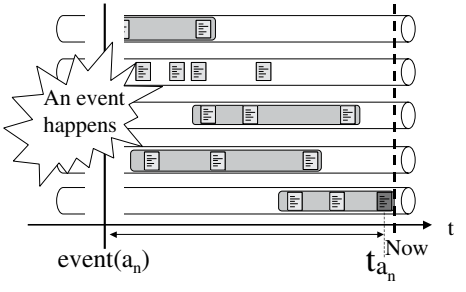


Fig. 8. Confidence level and Scoop level

where w_1 and w_2 are the weight value.

4 Carthage: A prototype system

4.1 System architecture

We have been developing a prototype system called *Carthage*. The system pulls articles from news servers at regular intervals. The server generates FAs from them, and calculates priority level based on temporal information and content similarity. The architecture of the system is shown in Fig.9. Carthagehas the following procedures.

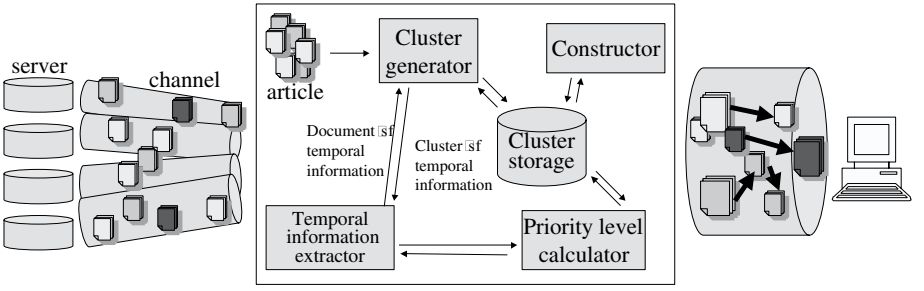


Fig. 9. System architecture

Cluster generator Articles are disseminated from the news source server at regular intervals. The server acquires the table-of-contents page from the news source server periodically. FAs are generated with time-series clustering mechanism. Newly arrived articles are clustered into existing FAs. When it belongs to neither of the clusters, it is stored as the first news article of a new topic.

Temporal information extractor The extractor filters the disseminated time[3] and the time when the event occurs actually.

Priority level calculator The priority level is calculated by the mechanism mentioned in Fig. 9. The priority level of stored articles are automatically calculated as time passes when a new article is added to the FA.

4.2 Reconstruction and presentation

Articles stored in the cluster storage are automatically distributed based on the priority level. If two articles which belong to the same FA are selected, only the latest article should be distributed. Articles with high confidence levels should be emphasized, and articles with high scoop levels should be disseminated in a preferential sequence. Carthage generates Flash contents from the articles. These contents are shown to the user on a ticker and pop-up text box which are automatically created from Macromedia GENERATOR[7] templates.

[3] Disseminated time is the transaction time when an article is stored strictly in the server.

5 Concluding Remarks

In this paper, we described a multi-channel dissemination system based on time-series clustering mechanism for on-line news articles. We proposed an algorithm for detection of follow-up article groups with time-series clustering mechanism, and we proposed a method of calculating a priority level based on delivery time and similarity. Also, we described a method for filtering on-line news articles based on priority level which is dynamically changing by the detection of follow-up articles. Furthermore, we described the implementation issues of the prototype system based on the proposal mechanism.

Acknowledgments

This research was supported by the Research for the Future Program of Japan Society for the Promotion of Science under the project "Researches on Advanced Multimedia Contents Processing".

References

1. Michael Franklin and Stan Zdonik. Push technology in perspective. In *Proc. of ACM SIGMOD'98 International Conference on Management of Data*, pages 516–519, 1998.
2. PointCastNetwork. http://www.pointcast.com/.
3. Yahoo! News. http://dailynews.yahoo.com/. Yahoo! Inc.
4. Hot channel. http://channel.goo.ne .jp/. NTT-X.
5. Lycos News. http://www.lycos.com/news/. Lycos, Inc.
6. CNN.com. http://www.cnn.com/. Cable News Network.
7. GENERATOR. http://www.macromedia.com/software/generator/.
8. Ma Qiang, Hiroyuki Kondo, Kazutoshi Sumiya, and Katsumi Tanaka. Virtual TV Channel: Filtering, Merging and Presenting Internet Broadcasting Channels. In *Proc. of ACM Digital Library Workshop on Organizing Web Space (WOWS)*, pages 32–43, 1999.
9. Yiming Yang, Tom Pierce, and Jaime Carbonell. A study in retrospective and on-line event detection. In *Proc. of International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 28–36, 1998.
10. James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Proc. of International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45, 1998.
11. Sujeet Pradan, Takeshi Sogo, Keishi Tajima, and Katsumi Tanaka. A New Algebraic Approach to Retrieve Meaningful Video Interval from Fragmentarily Indexed Video Shots. In *Proc. of Advances in Visual Information Management Visual Database Systems(VDB5)*, pages 11–30. Kluwer Academic Publishers, 1999.
12. Koichi Munakata, Masatoshi Yoshikawa, and Shunsuke Uemura. On synchronous properties of periodically generated data sequence. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE)*, pages 294–301, 1999.

Optimizing Queries in Extended Relational Databases

Michael Maher and Junhu Wang

CIT, Griffith University, Brisbane, Australia 4111
M.Maher@cit.gu.edu.au, jwang@cit.gu.edu.au

Abstract. We investigate the optimization of extended relational queries used in systems holding, for example, spatial, multimedia or constraint data. For such queries we must account for the built-in relations specific to the kind of data, and application dependent relationships between different relations. We show that the constraint database perspective and the use of constrained tuple-generating dependencies provides a general framework in which to address semantic query optimization for these queries. We establish some sufficient conditions for query transformations involving the introduction of relations, extending work in the literature for conventional databases. We introduce *semantic query partition (SQP)* as a useful technique for optimizing queries with expensive operations, and investigate the problem of generating subqueries, which is central to the use of SQP.

1 Introduction

Query optimization is highly successful for conventional relational databases. Current techniques are powerful and reliable enough to provide a reasonably efficient implementation of any query. Such optimization techniques can take into account common forms of application-dependent semantic information expressed as data dependencies.

However, this state of affairs does not hold for many modern extensions of relational databases, such as databases holding spatial or video data, and constraint databases. In such databases there are three issues that affect query optimization at the query language level:

- The query language contains semantic relations that are built in.
- The application-specific semantic information varies more widely in form.
- The query language contains expensive operations.

We will consider these issues in turn.

An expressive query language must provide some built in semantic relations that have fixed meaning, independent of the data. In conventional databases, arithmetic relations (e.g. $X < Y$ or $Z = X + Y$) are in this category. For spatial data, there are topological and directional relations between spatial objects, such as X *contains* Y or X *is northeast of* Y. For video data there are temporal

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 386–396, 2000.

relations and kinetic properties. In constraint databases, these relations are the constraints themselves, and they may appear in the data in addition to queries.

In addition to straightforward semantic information such as functional dependencies and referential integrity constraints, the semantic relationships in modern databases require more complex expression, often involving the fixed semantic relations. For television data, there may be relationships between the video, audio, and closed-caption elements of the data; there may also be annotations related to video content [BYY97]. For spatial data we may want to express, for example, that every state capital is contained in the appropriate state, or that whenever an easement intersects a property the corresponding title notes that fact.

As data is becoming more complex, the operations permitted in a query language are becoming more expensive. For spatial data, computing the area of an object, the objects that touch it or the distance to the nearest object are expensive. For video data, almost any selection based on video content is expensive. In constraint databases, selections or joins on constraint attributes involve expensive constraint solving. In all these cases, it pays to spend time exploiting semantic information since potential improvements are much greater than in conventional databases.

In this paper, we examine the problem of query optimization with reference to these issues. Conventional techniques, which are largely *not* semantic-based, are brittle and/or ineffective for many modern applications. We treat the problem uniformly, from a constraint database perspective: the built-in relations are constraints and we permit a large class of data dependencies (the constrained tuple-generating dependencies of [MS]) to express the application-dependent semantic relationships. We will focus first on determining correct (i.e. meaning-preserving) modifications of queries and then address practical approximations. We will not address the problem of assessing the costs of different queries.

Many of the issues enumerated above apply also to distributed data (where remote data is expensive, and semantic relationships describe the relation between local and remote data) and to heterogeneous databases where semantic relationships describe the relation between data in different databases). There is already related work in these contexts on similar issues, usually in terms of materialized views (for example, [Gry99,LMSS]). While our results might be useful in these contexts, we will not specifically address such databases.

Following some technical preliminaries in the next section, we look first at the addition of predicates to queries. We then introduce semantic query partitioning as a technique for avoiding expensive operations. Using this technique requires finding queries contained in the original query, and we develop some results to aid in performing this task. Finally, we briefly discuss future work.

2 Preliminaries

2.1 Conjunctive Formula and Containment Mapping

A **conjunctive formula** is a formula of the form

$$p_1, ..., p_s, c_1, ..., c_t$$

where each p_i is a relation in the database scheme (referred to as an **ordinary relation** or simply a **relation**); each c_j is a constraint in a fixed constraint domain (referred to as a **built-in constraint** or simply a **built-in**). We will use **predicate** to mean either a relation or a built-in.

Generally, there can be variables, constants, or functions in any attribute positions of a relation, they are called **terms**. If the attributes of the relations are limited to *distinct variables*, then the conjunctive formula is said to be in **normal form**. Clearly, any conjunctive formula can be easily put into normal form.

Let $F_i = P_i, C_i$ $(i = 1, 2)$ be two conjunctive formulas, where P_i is the set of relations and C_i is the set of built-ins. Let $Var(F_i)$ be the set of all free (i.e, not existentially quantified) variables in F_i. A **symbol mapping** from F_2 to F_1 is a function from $Var(F_2)$ to the terms of F_1. If δ is a symbol mapping from F_2 to F_1, then $\delta(F_2)$ $(resp.\ \delta(P_2), \delta(C_2), \delta(X)$, where X is a sequence or subset of variables in $F_2)$ denotes the conjunctive formula (resp. relations, built-ins, a sequence or set of variables and constants in F_1) obtained by simultaneously replacing each variable x in F_2 (resp. P_2,C_2, X) by $\delta(x)$. δ is called a **containment mapping** from F_2 to F_1 if $\delta(P_2) \subseteq P_1$. If there is a single containment mapping δ from F_2 to F_1 such that $C_1 \rightarrow \delta(C_2)$, then we say that F_2 **strongly subsumes** F_1 by δ.

A **conjunctive query** Q is of the form

$$\{O : F\}$$

where $F = P, C$ is a conjunctive formula, referred to as the **body** of query Q; O is a non-empty sequence of terms, referred to as the **output** of Q. In this paper, we will assume that O is a sequence of variables, and there are only variables and constants in any attribute positions of the ordinary relations. Obviously, every conjunctive formula can be easily put into this form. Q is said to be in **normal form** if its body is in normal form, O is a sequence of *distinct variables* which do not appear in any ordinary relations, and all free variables in the built-in constraints occur either in the output or in some ordinary relations.

Let $Q_i = \{O_i : F_i\}$ $(i = 1, 2)$ be two conjunctive queries. If a containment mapping δ from F_2 to F_1 turns O_2 to exactly O_1, then δ is said to be a **containment mapping** [Ull88] from Q_2 to Q_1.

Let Q_1 and Q_2 be any two queries (not necessarily conjunctive). We say that Q_2 **contains** Q_1, denoted as $Q_1 \sqsubseteq Q_2$, if the answer set for Q_1 is a subset of the answer set for Q_2 for every database instance. Similarly, we say that Q_1 is an empty query (denoted as $Q = \emptyset$) if its answer set is empty for all database instances.

2.2 Integrity Constraints

Common integrity constraints in a database can be expressed using data dependencies, such as functional dependencies, constraint generating dependencies (CGDs), tuple generating dependencies (TGDs) [BCW99,FV]. A more general data dependency, named **Constraint Tuple Generating Dependency (CTGD)**, is studied in [MS]. Formally, a CTGD is a formula of the form

$$p_1(X_1), ..., p_n(X_n), C(X) \rightarrow \exists_{Y_1,...,Y_m}\ q_1(X'_1, Y_1), ..., q_m(X'_m, Y_m), D(X, Y) \quad (1)$$

where $X = X_1 \cup ... \cup X_n; Y = Y_1 \cup ... \cup Y_m; X'_i \subseteq X; X \cap Y = \emptyset$; p_i, q_j are relations; $C(X)$ is a conjunction of built-in constraints with *free* variables from X, $D(X, Y)$ is a conjunction of built-in constraints with *free* variables from X and Y; The conjunctive formula at the left side of the arrow "$\rightarrow$" is called the **antecedent**, the formula at the right side is called the **consequent**. Some special-form CTGDs we will consider in this paper include

(1) Constraint-Generating Dependencies (CGDs):

$$p_1(X_1), ..., p_n(X_n), C(X) \rightarrow D(X)$$

(2) Nullity-Generating Dependencies (NGD):

$$p_1(X_1), ..., p_n(X_n), C(X) \rightarrow FALSE$$

(3) Tuple-Generating Dependencies (TGDs):

$$p_1(X_1), ..., p_n(X_n), C(X) \rightarrow \exists_{Y_1,...,Y_m}\ q_1(X'_1, Y_1), ..., q_m(X'_m, Y_m)$$

(4) Tuple-Generating Dependencies with Empty Built-ins (TGDEB) :

$$p_1(X_1), ..., p_n(X_n) \rightarrow \exists_{Y_1,...,Y_m}\ q_1(X'_1, Y_1), ..., q_m(X'_m, Y_m)$$

In what follows, when we say integrity constraint, we always mean a CTGD, either in the general or a special form. Furthermore, we will assume that the antecedent of every integrity constraint is in normal form (although in some examples we will use more compact forms).

Definition 1. *Given a set of integrity constraints $\triangle$ and two queries Q_1 and Q_2, we say that Q_1 and Q_2 are* **equivalent** *(resp. Q_1 is* **contained** *in Q_2; Q_1 is* **empty***)* **under** *$\triangle$, denoted as $Q_1 =_\triangle Q_2$ (resp. $Q_1 \sqsubseteq_\triangle Q_2$; $Q_1 =_\triangle \emptyset$) if their answer sets are the same (resp. the answer set for Q_1 is a subset of that for Q_2; the answer set for Q_1 is empty) for all database instances satisfying $\triangle$.*

3 Adding Predicates to Conjunctive Queries

We can use integrity constraints to generate predicates to add to a query. In general, this transformation produces a union query.

Theorem 1. *Let $\triangle = \{P_i, C_i \rightarrow \exists_{Y_i}\ S_i, D_i \mid i = 1, ..., n\}$ be a set of CTGDs as defined in Section 2.2, where S_i is the set of relations in the consequent. Let a conjunctive query be $Q = \{O : R, E\}$, where R is the set of relations, and E is the set of built-ins. If there exist containment mappings $\delta_{i,1}, ..., \delta_{i,k_i}$ $(i = 1, ..., n)$ from P_i, C_i to R, E such that*

$$E \rightarrow \bigvee_{i=1}^{n} \bigvee_{j=1}^{k_i} \delta_{i,j}(C_i) \tag{2}$$

then $Q =_{\triangle} \bigcup_{1 \leq i \leq n, 1 \leq j \leq k_i} \{O : R, E, \delta_{i,j}(S_i, D_i)\}$, where all existential variables in $\delta_{i,j}(S_i, D_i)$ are replaced by distinct variables not previously occurring in Q.

Example 1. Let $\triangle = \{ic1, ic2\}$ and Q be as follows

$ic1: \quad r_1(x, y), x < y \rightarrow x = 0.$

$ic2: \quad r_2(x, y), x < y \rightarrow y = 0.$

$Q: \quad \{x, y : r_1(x, y), r_2(y, z), x < z\}$

where x, y, z are all from the reals.

Then $\delta_1 : x \rightarrow x, \quad y \rightarrow y \quad$ *and* $\quad \delta_2 : x \rightarrow y, \quad y \rightarrow z$ are containment mappings from the antecedents of $ic1$ and ic_2 respectively to the body of Q. Furthermore, $x < z \rightarrow \delta_1(x < y) \vee \delta_2(x < y)$. By Theorem 1, $Q =_{\triangle} \{x, y : r_1(x, y), r_2(y, z), x = 0, z > 0\} \cup \{x, y : r_1(x, y), r_2(y, z), x < 0, z = 0\}$.

The transformation in the above example cannot be obtained using traditional semantic query optimization approaches.

If there are no built-ins (i.e, the built-ins are all TRUE) in the antecedents of the integrity constraints, then the condition in Theorem 1 reduces to the mere existence of a single containment mapping, because the implication relationship (2) is trivially true for any such mapping. Thus Theorem 1 indicates the significance of built-in constraints in the antecedent of an integrity constraint. It also generalizes previous results on restriction and join introduction.

In order to use Theorem 1 to add new predicates to a conjunctive query, we need to find the relevant containment mappings. Obviously, if a conjunctive formula is in normal form, then it is easy to enumerate all containment mappings from it to another conjunctive formula. The biggest difficulty in using Theorem 1 lies in testing the implication of built-in constraints. However, in most practical cases, there are only a few integrity constraints (each of which has only a few built-ins), and the number of relevant containment mappings is very limited, so the constraint implication problem can be checked rather quickly. Thus we may find the transformation in Theorem 1 still practically useful, especially when the added predicates can "filter out" some costly condition tests which are frequently met in extended relational databases.

4 Semantic Query Partition

When a query Q can be partially answered by another one Q' (i.e, $Q' \sqsubseteq Q$), and Q' is considerably cheaper than Q, it is sometimes beneficial to divide the

evaluation of Q into three steps: First find all answers for Q' and all tuple combinations that produce these answers; then check the tuple combinations that are left for other answers to Q; finally return the union of the results in the previous two steps. The benefit of this partition is particularly evident when Q has expensive operations which are not present in Q', Q' contains a large part of all answers to Q, and there are a large number of tuple combinations in the base relations satisfying the conditions of Q. This idea has been used in, for instance, spatial databases, where when we want to find all objects that are not contained in a given area, we first use the minimum bounding boxes (MBB) to find part of the answer, and only for those objects whose MBBs are contained in the MBB of the area do we actually check the containment using a more complicated algorithm. Similar ideas have also been used in situations such as when materialized views or cached results are available [CKPS95,Qia,Gry99].

In traditional databases, the difference between the costs of various restriction or join conditions are relatively small, therefore the improvement (if any) in query efficiency due to the partition is most likely to be insignificant. However, this is not so when expensive predicates are involved.

In this section, we consider the problem of using integrity constraints to partition the processing of conjunctive queries into the three steps as described above, which we refer to as **semantic query partition (SQP)**. The aim of the partition is to reduce the number of tests on expensive conditions. We start with a motivating example.

Example 2. Suppose we have the following relations:
$state(stateName, stateGeometry)$,
$own(ownerName, landType, landGeometry)$,
$fa(memberID, personName, registerState)$,

where *stateGeometry* and *landGeometry* are spatial attributes, and all other attributes are alphanumeric. The relation fa represents data about members of the Farmers Association.

Suppose we have the following integrity constraint:

Any member of the farmers association has some farm land ($landType = 1$) in his/her registration state, that is
$fa(id, pn, sn) \rightarrow \exists_{lg,sg}\ state(sn, sg), own(pn, 1, lg), lg \cap sg \neq \emptyset$

Consider the query Q: Find all people who own some farm land in state CA.
$Q = \{pn : own(pn, 1, lg), state(CA, sg), lg \cap sg \neq \emptyset\}$.

By the integrity constraint, the answer includes all members of the farmers association who registered in CA, thus we need check only the remaining land owners to see whether they too own some farm land in CA. That is, we can partition Q into $Q' = \{pn : fa(id, pn, sn), sn = CA\}$ and $Q - Q' = \{pn : (fa(id, pn, sn), sn \neq CA, own(pn, 1, lg)), state(CA, sg), sg \cap lg \neq \emptyset\}$. Obviously Q' is a much cheaper query than Q, therefore, if most farm owners in CA are registered in the farmers association and there are many people who own some farm land in CA, then we can save a lot of work by the partition.

4.1 Semantic Query Containment

The SQP problem is closely related to the semantic query containment problem (SQC) –determining whether a conjunctive query is contained in another (or the union of some other conjunctive queries) under given integrity constraints . Thus we first discuss the latter. As by-products of the results here, the empty query detection (under given integrity constraints) problem is also addressed.

Let $Q_0, Q_1, ..., Q_k$ be conjunctive queries, and $\triangle$ be a set of CTGDs. Intuitively, if Q_0 is not contained in the union of $Q_1, ..., Q_k$ without $\triangle$, but $Q_1 \sqsubseteq Q_2$ when $\triangle$ is imposed, then the queries and integrity constraints must be related in some way. Our next theorem says the existence of some particular containment mappings is such a necessary relationship.

Theorem 2. *Let $Q_l = \{O_l : R_l, E_l\}$ (for $l = 0, 1, ..., k$) be conjunctive queries. For $0 < l \leq k$, Q_l is in normal form, Q_0 is not contained in $\bigcup_{l=1}^{k} Q_l$. Let $\triangle = \{P_i(X_i), C_i(X_i) \rightarrow S_i, D_i | i = 1, ..., n\}$ be a set of CTGDs. If $Q_0 \sqsubseteq_{\triangle} \bigcup_{l=1}^{k} Q_l$, then*

(1) there must be containment mappings from some $P_i(X_i), C_i(X_i)$ to R_0,E_0.

(2) Suppose $\delta_{i,1}, ..., \delta_{i,k_i}$ are all containment mappings from $P_i(X_i), C_i(X_i)$ to R_0, E_0 (for $i = 1, ..., n$), and $\rho_{l,1}, ..., \rho_{l,s_l}$ are all containment mappings from R_l, E_l to R_0, E_0 (for $l = 1, ..., k$), then $E_0 \rightarrow \vee_{i=1}^{n} \vee_{j=1}^{k_i} \delta_{i,j}(C_i(X_i)) \vee \vee_{l=1}^{k} \vee_{t=1}^{s_l} \rho_{l,t}(E_l)$.

Theorem 2 provides a necessary condition for the SQC problem. The next theorem gives a condition that is both necessary and sufficient when the integrity constraints are all NGDs.

Theorem 3. *Let $N = \{P_i, C_i \rightarrow FALSE \mid i = 1, ..., n\}$ be a set of NGDs, $Q_l = \{O_l : R_l, E_l\}$ $(l = 0, 1, ..., k)$ be queries, $Q_0 \neq_N \emptyset$, and for $l > 1$, Q_l is in normal form. Let $\delta_{i,1}, ..., \delta_{i,k_i}$ be all containment mappings from P_i, C_i to R_0, E_0. Then $Q_0 \sqsubseteq_N \bigcup_{l=1}^{k} Q_l$ iff there are containment mappings $\rho_{l,1}, ..., \rho_{l,s_l}$ from Q_l to Q_0 such that*

$$E_0 \rightarrow \vee_{i=1}^{n} \vee_{j=1}^{k_i} \delta_{i,j}(C_i) \vee \vee_{l=1}^{k} \vee_{t=1}^{s_l} \rho_{l,t}(E_l).$$

A necessary and sufficient condition can also be established when the integrity constraints consist of some NGDs and a set of non-recursive TGDEBs. To do this, we need the concept of *semantic expansion* of a query.

Definition 2. *Given a query $Q = \{O : R, E\}$, and a set S of CTGDs, the* **semantic expansion of Q under** *S, denoted as Q^S, is a query $\{O : R^S, E^S\}$ such that*

1. $R \subseteq R^S$

2. *For any integrity constraint* $P, C \rightarrow \exists_Y\ q_1, ..., q_m, D$ *in* S*, if* P, C *strongly subsumes* R^S, E *by a containment mapping* δ*, then* $\delta(q_i) \in R^S$ $(i = 1, ..., m)$ *(where all existential variables in* $\delta(q_i)$ *have been properly renamed), and* $E^S \rightarrow \delta(D)$.
3. E^S *is the weakest set of built-ins that satisfy the previous condition.*

If the given CTGDs are not recursive (that is, S does not contain a sequence of CTGDs ic_i $(i = 1, ..., n)$ such that some relation names in the antecedent of ic_{i+1} appear in the consequent of ic_i for $i = 1, ..., n-1$, and some relation names in the antecedent of ic_1 appear in the consequent of ic_n), then Q^S contains a finite number of predicates, and it can be computed by repeatedly considering each integrity constraint against Q^S (initially, $Q^S = Q$) until no more predicates can be added to Q^S except duplicates.

Theorem 4. *Let* Q *be a conjunctive query,* Q' *be any query. Let* N *be a set of NGDs,* S *be a set of non-recursive TGDEBs. Then* $Q \sqsubseteq_{N,S} Q'$ *iff* $Q^S \sqsubseteq_N Q'$.

Theorem 4 indicates that as far as SQC is concerned, the TGDEBs and the NGDs can be considered in sequence, that is, once we have computed Q^S, S can be discarded.

The above result can be extended to TGDs when the buit-in constraints have the Independence of Negative Constraints (INC) property (see [LM92] for the definition of INC). However, when the built-ins do not have INC, the theorem does not hold if S is a set of TGDs with non-empty built-ins, as shown in the following example. Thus, we have established the limits of the semantic expansion technique for such problems.

Example 3. Let $Q = \{x, y : p(x, y)\}$ and let Q' be an empty query. Let S contain $p(x, y), c(x, y) \rightarrow q(x, y)$, where $c(x, y)$ is any non-trivial constraint (not TRUE or FALSE). Let N contain $p(x, y), \neg c(x, y) \rightarrow FALSE$ and $p(x, y), q(x, y) \rightarrow FALSE$.
The only containment mapping from $p(x, y), c(x, y)$ to $p(x, y)$ is $\delta : x \rightarrow x, y \rightarrow y$, but $TRUE$ does not imply $c(x, y)$, so $Q^S = Q$. It is easy to see that Q^S is not empty under N, but Q is empty under N and S.

The next corollary follows from Theorem 3 and Theorem 4.

Corollary 1. *Let* $Q_l = \{O_l : R_l, E_l\}$ $(l = 0, 1, ..., k)$ *be conjunctive queries.* $Q_1 \neq_{N,S} \emptyset$*. Let* $N = \{P_i, C_i \rightarrow FALSE\}$ *be a set of NGDs and* S *be a set of non-recursive TGDEBs. Let* $\delta_{i,1}, ..., \delta_{i,k_i}$ *be all containment mappings from* P_i, C_i *to* R_0^S, E_0^S*. Then* $Q_0 \sqsubseteq_{N,S} \bigcup_{l=1}^{k} Q_l$ *iff there are containment mappings* $\rho_{l,1}, ..., \rho_{l,s_l}$ *from* Q_l $(l = 1, ..., k)$ *to* Q_0^S *such that*

$$E_0^S \rightarrow \vee_{i=1}^{n} \vee_{j=1}^{k_i} \delta_{i,j}(C_i) \vee \vee_{l=1}^{k} \vee_{t=1}^{s_l} \rho_{l,t}(E_l).$$

4.2 Query Emptiness

The problem of detecting query emptiness is a special case of query containment, where we take the containing query to be an empty query. Thus the work above on query containment can be applied directly to the problem of query emptiness. For example, the following is an immediate corollary of Corollary 1.

Corollary 2. *Given a non-empty conjunctive query $Q = \{O : R, E\}$, a set $N = \{P_i, C_i \rightarrow FALSE \mid i = 1, ..., n\}$ of NGDs, and a set S of non-recursive TGDEBs, we have $Q =_{N,S} \emptyset$ if and only if there are containment mappings $\delta_{i,1}, ..., \delta_{i,k_i}$ from P_i, C_i to R^S, E^S such that*

$$E^S \rightarrow \vee_{i=1}^{n} \vee_{j=1}^{k_i} \delta_{i,j}(C_i).$$

[ZÖ] gives two theorems which are special cases of Corollaries 1 and 2, with S being a set of *referential integrity constraints* (TGDEBs with a single relation in the antecedent and a single relation in the consequent). Our results extend theirs, and Example 3 shows that our results are unlikely to be extended much further.

4.3 Finding a Subquery

Definition 3. *Let $\triangle$ be a set of integrity constraints, Q and Q' be two conjunctive queries. If*

(1) $Q' \sqsubseteq_{\triangle} Q$

(2) There are predicates[1] *in Q that are not present in Q'*

Then we say Q' is a **subquery** *of Q under $\triangle$.*

The key of SQP is to find a subquery Q' of Q. Whether it is worthwhile to use Q' in SQP has to be decided by a cost model.

Let the conjunctive query Q be $\{O : R, E\} = \{O : R_1, R_2, E_1, E_2\}$, where $R = R_1, R_2$ are the relations and $E = E_1, E_2$ are the built-ins. Let E_2 be the expensive restrictions we want to avoid. We require that Q be in normal form so that all the expensive join conditions will be in E.

There are many different types of integrity constraints, with some of which, the task of finding a subquery of Q under them is trivial (e.g, if $P(Z) \rightarrow R(Z), E(Z)$ then $\{O : P(Z)\} \sqsubseteq \{O : R(Z), E(Z)\}$). We cannot enumerate all the possibilities. Therefore, we will only mention two typical cases.

Case A The integrity constraints imply $R, R', E', E_1 \rightarrow E_2$, where R' is a set of relations other than those in R, E' is a set of built-ins that are not in E_2. In this case, $Q' = \{O : R, R', E', E_1\}$ is a subquery of Q. In Q', we don't need to test the conditions specified by E_2, but we have added some relations R' to be joined and some other restrictions E'.

[1] Intuitively, these are the expensive operations we want to avoid as much as possible.

Case B The integrity constraints imply $R', E', E_1 \rightarrow R, E_2,$, where R', E' are the same as in the first case.
In this case, if $O \subseteq Var(R')$ then $\{O : R', E', E_1\}$ is a subquery of Q.

To test whether the conditions in the above cases hold under $\triangle$, we can use various chasing procedures such as those in [MS]. The problem is that we often don't know what R' or E' is in advance. However, sometimes a single (or more) integrity constraint(s) in $\triangle$ is applicable simply by variable renaming. These cases are summarized in the following propositions $1 \sim 2$.

Let P, C be a conjunctive formula, Z be a set of terms. For any mapping δ from a subset X' of $Var(P, C)$ to Z, we will use $\delta(P(X), C(X))$ to denote the formula obtained from $P(X), C(X)$ by simultaneously replacing all variables in X' by their images under δ.

Definition 4. *A containment mapping δ from P, C to R, E is said to be a* **full mapping** *if $\delta(P) = R$.*

Proposition 1. *Let $\triangle = \{P_i(X_i), C_i(X_i) \rightarrow D_i(X_i) \mid i = 1, ..., n\}$ be a set of CGDs. Let $Q = \{O : R(Z), E_1(Z), E_2(Z)\}$. If there are full mappings $\delta_{i,1}, ..., \delta_{i,k_i}$ $(i = 1, ..., n)$ from $P_i(X_i)$ to $R(Z)$ such that $\wedge_{i=1}^{n} \wedge_{j=1}^{k_i} \delta_{i,j}(C_i(X_i)) \neq E_2(Z)$, and $\wedge_{i=1}^{n} \wedge_{j=1}^{k_i} \delta_{i,j}(D_i(X_i)) \wedge E_1(Z) \rightarrow E_2(Z)$, then $\{O : R(Z), E_1(Z), \wedge_{i=1}^{n} \wedge_{j=1}^{k_i} \delta_{i,j}(C_i(X_i))\}$ is a subquery of Q under $\triangle$.*

Proposition 1 corresponds to a sub-case of Case A. The next example shows an application of it.

Example 4. Let the integrity constraints and the query be as follows
ic1: $r_1(x, y), r_2(y, z) \rightarrow x \neq \emptyset$
ic2: $r_1(x', y'), r_2(y', z'), y' > 0 \rightarrow x' \subseteq z'$
Q: $\{u, w : r_1(u, v), r_2(v, w), v < 10, u \cap w \neq \emptyset\}$
Define $\delta_1 : x \rightarrow u,\ y \rightarrow v,\ z \rightarrow w$ and $\delta_2 : x' \rightarrow u, y' \rightarrow v, z' \rightarrow w$, then δ_1 and δ_2 are full mappings from $r_1(x, y), r_2(y, z)$ to $r_1(u, v), r_2(v, w)$ and from $r_1(x', y'), r_2(y', z')$ to $r_1(u, v), r_2(v, w)$ respectively. Furthermore, $\delta_1(x \neq \emptyset) \wedge \delta_2(x' \subseteq z') \rightarrow u \cap w \neq \emptyset$, thus, $Q1 = \{u, w : r_1(u, v), r_2(v, w), v > 0, v < 10\}$ is a subquery of Q under $ic1, ic2$.

The next proposition corresponds to Case B.

Proposition 2. *Let ic be a CTGD of the form $P(X), C(X) \rightarrow \exists_Y S(X', Y), D(X', Y)$ $(Y \cap X = \emptyset)$. If there is a full mapping δ from $S(X', Y)$ to $R(Z)$, such that*
(1) $Var(\delta(X)) \cap Var(\delta(Y)) = \emptyset$,
(2) $\delta(D(X', Y)) \rightarrow E_2(Z)$,
(3) $O \subseteq \delta(X)$, and $Var(E_1(Z)) \subseteq \delta(X)$,
(4) $\delta(C(X)) \neq E_2(Z)$,
then $\{O : \delta(P(X)), \delta(C(X)), E_1(Z)\}$ is a subquery of Q under ic.

Example 2 is a case where Proposition 2 can be applied.

5 Conclusion and Future Work

We have addressed the problem of semantic query optimization for extended relational databases. For spatial, multimedia and constraint data, the presence of expensive operations makes semantic query optimization more cost-effective than in conventional databases. Based on a more thorough analysis of the built-in constraints and interactive effects of several integrity constraints, our results generalize previous existing works on predicate introduction, emptiness detection, and query containment. We introduced semantic query partitioning as a potentially useful transformation when expensive predicates are present.

However, whether the newly obtained query is better than the original one must be decided by a cost model. That means the semantic transformation methods provided here must be combined with a traditional query optimizer. In addition, to reduce the cost of semantic transformation, constructive algorithms need to be designed that can directly build a semantically equivalent query with guaranteed better performance. Efficient constraint solvers are imperative for the transformations to be practical. Such solvers existing, they can be embedded into a semantic query optimizer. All of these fall into our future work.

References

[BCW99] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. *Journal of Computer and System Sciences 59, 94-115*, 1999.

[BYY97] R. M. Bolle, B.-L. Yeo, and M. M. Yeung. Video query and retrieval. In *LNAI, 1342, 13-24*. Springer, 1997.

[CKPS95] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *International Conference on Database Engineering*, pages 190–200. IEEE, 1995.

[FV] R. Fagin and M. Y. Vardi. The theory of data dependencies—an overview. In *LNCS v172,1-22, Springer-Verlag, 1984*.

[Gry99] J. Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. *Information Systems, 24(7): 579-612*, 1999.

[LM92] Jean-Louis Lassez and Ken McAloon. A canonical form for generalized linear constraints. *Journal of Symbolic Computation, 13(1), 1-24*, 1992.

[LMSS] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. ACM Symp. PODS, 14: 95-104*, New York, 1995.

[MS] M. J. Maher and D. Srivastava. Chasing constrained tuple-generating dependencies. In *Proc. ACM Symp. PODS, New York, 1996*.

[Qia] X. Qian. Query folding. In *Proc. 12th International Conference on Data Engineering, 48-55, Louisiana, Feb 26 - Mar 1, 1996*.

[Ull88] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1 & 2. Computer Science Press, 1st edition, 1988.

[ZÖ] X. Zhang and Z. M. Özsoyoglu. Implication and referential constraints: A new formal reasoning. *IEEE TKDE, 9(6):894-910, Nov/Dec 1997*.

Reducing the Location Query Cost Based on Behavior-Based Strategy

Ming-Hui Jin, Jorng-Tzong Horng, Hsiao-Kwang Wu, and Baw-Jhiune Liu

Department of Computer Science and Information Engineering
National Central University, Taiwan
{jinmh, horng, hsiao, bjliu}@db.csie.ncu.edu.tw

Abstract. In a location query process the most expensive procedure is maintaining the location information of mobiles. To reduce the expensive cost, we adopt the profile-based approach to design a behavior-based strategy (BBS) based on the moving behavior of each mobile generated by long-term collection of its moving history. We use a data mining technique to mine the moving behavior of each mobile and then estimate the probability that each mobile stays in each location at each time region given the last known location from its moving behavior. To reduce unnecessary computation, we consider the location tracking and computational cost and then derive a cost model. A greedy heuristic is proposed to minimize the cost model through finding the appropriate checkpoints. The experimental results show our strategy outperforms fixed paging area strategy currently used in GSM or IS-54 system and time-based strategy for highly regular moving mobiles.

1. Introduction

Under the sharply developing wireless communication technologies, more and more mobile stations will access databases over the wireless communication channels. Under this situation more and more mobiles will become the data source for future databases, or furthermore, the mobile themselves become the information that is allowed to be queried by some other clients. The fact that clients and servers in this mobile environment can change locations enables the possibility of making and answering queries in a way that is dependent on the current position of the queried data if the queried information are attributes of some mobile objects.

When a database system is required to process a location query, it needs paging procedures to capture the exact location of all mobiles that concern the location query. Because the unit cost to page a mobile is more expensive than other procedures of a location query due to the precious radio bandwidth resource, the location tracking cost minimization becomes the main issue for location query problem. The fixed paging area strategy currently used by second generation PCS is designed as follows [1-5]. All base-stations are partitioned into several registration areas and then the paging area function is defined to be the registration area the mobile lastly updated. No location update is required if the mobile does not move to another registration area. This approach could be simply implemented but could not reach the optimal location

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 397–406, 2000.

tracking cost since the paging area defined in this approach may be too large to waste paging cost (for low mobility mobiles) or too small (for high mobility mobiles) to make unnecessary updates.

To reduce location query cost furthermore, several location tracking strategies in the literature have been proposed. Xie, Goodman and Tabbane [6, 7] introduced the individual location area concept that treats mobiles with different mobility and call characteristics differently to reduce the average signaling cost of mobility management. Based on this concept, several approaches such as time-based strategy [8-10], distance-based, movement-based strategy [11, 12] and profile-based strategy [13] were introduced successively.

Most of the above strategies define their paging area function according to short-term observation of each mobile. For example, the approach in [10] estimates the time-varying mobility pattern according to the (short-term) history of each PCS subscriber's location. Predict the most possible area the mobile will stay in the next time region and then define it to be the paging area of the next time region. This approach does reduce the update cost for high mobility mobiles, but the paging area increases dramatically with its moving velocities and it still need to updates its paging area periodically to PCS to keep the hit rate for paging procedure, even the object is immobile. The profile-based strategy proposed by Sami Tabbane [13] tries to reduce the location tracking cost by taking advantage of most mobiles' highly predictable patterns. Although the performance of this strategy is much better than the fixed paging area strategy currently adopted by most PCS, several important parameters such as the time-varying probabilities and the approach to partition each mobile moving period are innocent.

In this paper, we extend the profile-based strategy to develop a behavior-based strategy. We first conceptually introduce the behavior-based strategy (BBS) in Section 2. To define the paging area function, we define the moving behavior of each mobile and then adopt current data mining techniques such as association rules and Bayesian networks to mine the moving behavior from the long-term history of location change events in Section 3. To realize the BBS, all the necessary parameters and functions are defined, analyzed and estimated in Section 4. Experiments and comparison are presented in Section 5. Conclusion and future work are drawn in Section 6.

2. The Behavior-Based Strategy

The Behavior-Based Strategy (BBS) is designed to reduce the location updating cost of each mobile by taking advantage of the PCS subscriber's highly predictable patterns. We consider that the system handles a profile for each mobile. In such condition each mobile keeps a copy of its paging area function.

In this paper, we assume that the moving behavior of each mobile are regular enough to be mined from its moving history. In the following subsections we describe the paging and location update procedures of BBS.

2.1 Paging Procedure

The paging procedures of BBS are the same with the fixed paging area strategy except the data base query procedure and their query results. The query result of BBS is a set of base stations that compose the paging area at that time and the query result of fixed paging area is a registration area ID. Fig. 1 and Fig. 2 show the location database query procedures for these two strategies, respectively.

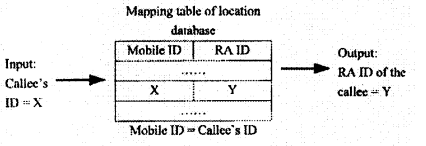


Fig. 1. Query procedure of fixed paging area strategy.

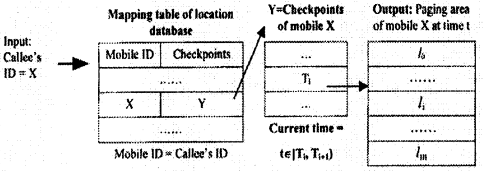


Fig. 2. The query procedure of BBS.

2.2 Location Update Procedure

In BBS, a mobile is required to hold a location updating procedure at time t whenever it leaves its paging area at time t. To reach this goal, each mobile needs to check whether it is in its paging area or not by querying its paging area function whenever the checkpoint time is up or it perceives that it moves to another cell. Fig. 3 shows the decision procedure of BBS.

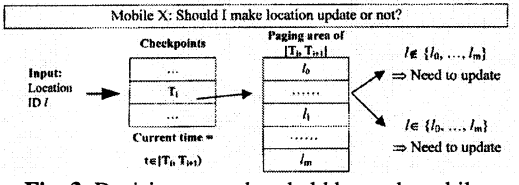


Fig. 3. Decision procedure held by each mobile.

If a mobile changes its location and the new location the mobile stays is still a member of PA(T_i), then the mobile does not need to make any location update. But if the new location is not inside PA((T_i), then the mobile should update its location

We design the rule of updating as follows. (Assume that the PCS has partition all its location into several registration areas.)

R1. Whenever the mobile changes its location from l_x to l_y at time $t \in (T_i, T_{i+1})$ and both l_x and l_y are elements of PA(T_i), then no location update is necessary to be made.

R2. Whenever the mobile changes its location from l_x to l_y at time $t \in (T_i, T_{i+1})$ where $l_x \notin PA(T_i, T_{i+1})$, the mobile needs not make location update if the two

locations l_x and l_y are in the same registration area. If l_y is not in the registration area that contains l_x, then

i. $PA(t)$ = The registration area that contains ly if $l_y \notin PA_0(t)$.

ii. $PA(t) = PA_0(t)$ if $l_y \in PA_0(t)$.

R3. At checkpoint time T_{i+1}, if the mobile stays in $l_x \in PA_0(t) \cap PA_0(T_{i+1})$ where $t \in (T_i, T_{i+1})$ is the last location updated time, then no location update is necessary and the paging area is renewed to be $PA_0(T_{i+1})$.

R4. At checkpoint time T_{i+1}, if the mobile stays in $l_x \in PA_0(t)$-$PA_0(T_{i+1})$ where $t \in (T_i, T_{i+1})$ is the last location updated time, then the mobile is required to make location update and $PA(T_{i+1})$ is set to be a registration area that contains l_x.

3. The Moving Behavior

3.1 Description of Moving Behavior

Based on the observation of many mobiles, we assume that a regular moving mobile often chooses one of a few paths it often passes through and the arrive time to each location on each path are like. Thus a moving behavior of a mobile should consist of several paths forms a partial ordered set. Although the time the mobile arrives to each location is not always the same, in most cases it forms a multi-modal distribution. To simplify our estimation procedure, the multi-modal distribution needs to be divided into several uni-modal distributions if the modals are not close. Here we assume that each uni-modal time distribution is in normal distribution. Thus, each vertex of a moving behavior should be an ordered pair (l, t) where l is the location the mobile often pass through and t is a normal distribution of the time the mobile arrived l.

Let $<(l_1, t_1), (l_2, t_2), (l_3, t_3), (l_4, t_4), \ldots, (l_n, t_n)>$ be a moving log of a mobile and D be its moving period, then we define the moving behavior of the mobile to be a partial ordered set of time-location information $<V, E, C_E>$ as follow:

- $V \subseteq \{(l, t, s) \in L \times T \times N\}$ where $L = \{ l_i \mid 1 \le i \le n \}$, T is the set of all normal distributions and s is the support of this vertex. The support of the vertex (l, t, s) with $t = N(\mu, \sigma)$ is defined to be the size of the set $\{(l_i, t_i) \mid l_i = l$ and $t_i - kD$ close to μ if $t_i - kD \in [0, D]\}$.
- $E \subseteq V \times V$. For each $(v_i, v_j) \in E$, there should exist enough supports (l_i, t_i) from the moving logs where (l_i, t_i) is a support of v_i and (l_{i+1}, t_{i+1}) is a support of v_j.
- For each $e = (v_i, v_j) \in E$, $C_E(e)$ is defined to be the confidence that the mobile will arrive v_j given the mobile is in v_i.

3.2 Algorithm to Mine the Moving Behavior from Moving Logs

To mine the moving behavior from a given moving log $<(l_1, t_1), \ldots, (l_n, t_n)>$, we first separate it into M logs in which the i^{th} log $Lg_i = <(l_{i,1}, t_{i,1}), \ldots, (l_{i,n(i)}, t_{i,n(i)})>$ presents the i^{th} moving period's moving log. Without loss of generality, we assume that $t_{i,j} \in [0, D]$ for all i and j. And then, for each $l \in L$, cluster the set $\{(l_{i,j}, t_{i,j}) \mid l_{i,j} = l\}$ into several

clusters $\{C_{l,j}\}$ according to the distribution $\{t_{i,j} \mid l_{i,j} = l\}$ and denote V to be the set $\{C_{l,j} \mid l \in L\}$. There are many database systems that provide clustering functions. For example, the DB2® Intelligent Miner for Data. In this paper we use SPSS to generate the clusters for each location l.

Some nodes that have less support should be removed from the moving behavior. The amount of the threshold should be in proportion to the amount of moving periods of the moving log M. In this paper we use M/4 to be the threshold in our experiments.

After we establish the vertex set V of the moving behavior, and the mean and the variance of each vertex $C_{l,j}$ of V, we construct the edges between vertices of V as follows. Let $S(C_i, C_j)$ to be the support of the ordered pair (C_i, C_j) and $S(C_i)$ to be the support of C_i.

$S(C_i, C_j) = 0$ for all i and j.
for(x = 1; x <= M; x++)
for(y = 1; y < n(x); y++)
{
Find C_i and C_j from V satisfy
1. $d(C_i, (l_{x,y}, t_{x,y})) <= d(C_k, (l_{x,y}, t_{x,y}))$ for all $C_k \in V$.
2. $d(C_j, (l_{x,y+1}, t_{x,y+1})) <= d(C_k, (l_{x,y+1}, t_{x,y+1}))$ for all $C_k \in V$.
Increase the support of (C_i, C_j).
}.

For each ordered pair (C_i, C_j), if $S(C_i, C_j)/S(C_i) > C_E$ then a directed link $C_i \rightarrow C_j$ is established.

C_E is the threshold for link generation and $C_E(C_i, C_j)$ is the confidence of the edge (C_i, C_j). For each $C_Z \in V$, we define $d(C_Z, (l_{x,y}, t_{x,y}))$ as follow:

1. $l_Z \neq l_{x,y} \Leftrightarrow d(C_Z, (l_{x,y}, t_{x,y})) = \infty$
2. $l_Z = l_{x,y} \Leftrightarrow d(C_Z, (l_{x,y}, t_{x,y})) = | \mu_Z - t_{x,y} | < \infty$.

4. Designing the Paging Area Function

4.1 Estimating the Time-Varying Location Probability

Cost Model

For each mobile, we define the paging area at time t to be a set of locations (base-stations) in which the PCS tries to find the mobile by paging this area at time t, and denote it as PA(t). Both PCS and mobiles are required to adjust the paging area whenever a location update is made.

The location tracking cost is defined to be the update cost and paging cost. In time region (T_i, T_j) the update cost could be defined as the number of location updates and the paging cost could be defined as $\sum_{i=1}^{k} PA(t_i)$ if there are k calls arrived at time $t_1, t_2, \ldots, t_k \in (T_i, T_j)$. In this paper, we assume that the call arrival rate is the same for all time t and denote it as λ.

Define STAY(A(t_i, t_j)) to be the probability that the mobile stays in the location area A(t) for all t ∈ (t_i, t_j) and define CALL(t_i, t_j) to be the number of calls arrive in some time of (t_i, t_j). If the time region (t_i, t_j) is short enough, then we could assume that there is at most 1 location update event and at most 1 call arrives in the time region, and we could also assume that the size of paging area in this time region has no change. Thus, the location update cost and paging cost in a short enough time region (t_i, t_j) could be defined as follows:

Location update cost = 1 - STAY(A(t_i, t_j)) **(1)**

Paging cost = CALL(t_i, t_j)*|(PA(t)| **(2)**

Where |(A)| is the number of elements of finite set A. And then we denote the density function stay(A(t)) and call(t) in (3) and (4), respectively.

$$stay(A(t)) = \lim_{\Delta t \to 0} STAY(A(t, t+\Delta t)) \quad \text{and} \tag{3}$$

$$call(t) = \lim_{\Delta t \to 0} \frac{CALL(t, \Delta t)}{\Delta t} \tag{4}$$

By definition of call arrival rate, and our assumption about call arrival rate, we have call(t) = λ. Thus we have the location tracking cost in time region [T_i, T_j] is

$$\int_{T_i}^{T_j} [(1 - stay(PA(t)) + \lambda \times sizeof(PA(t))]dt \tag{5}$$

Due to the limitation of this paper, we state the estimation of stay(PA(t)) as follows without proof.

$$\text{stay(PA(t))} = \sum_{l_i \in PA(t)} Path(l_i, l_j) \times CP(l_i, t) \tag{6}$$

Where

$$\text{Path}(l_i, l_j) = \sum_{\text{All path from } v_j \text{ to } v_i} \left(\prod_{y=1}^{x-1} Ce(v_{j_y}, v_{j_{y+1}})\right) \tag{7}$$

4.2. Decision the Initial Paging Area for Each Time Interval

In time region [T_i, T_{i+1}] we recompose the set of all locations into the sequence $l_{i,1}$, $l_{i,2}$, …, $l_{i,m}$ satisfies

$$\frac{1}{(T_{i+1} - T_i)|l_{i,j}|} \int_{T_i}^{T_{i+1}} P(l_{i,j}, t)dt \geq \frac{1}{(T_{i+1} - T_i)|l_{i,j+1}|} \int_{T_i}^{T_{i+1}} P(l_{i,j+1}, t)dt \quad \text{for all } j \geq 1 \tag{8}$$

In this way, the initial paging area $PA_0(T_i) = \{ l_{i,j} | 1 \leq j \leq k_i \}$ for some integer k_i. Whenever k_i is decided, the expected location tracking cost in time region (T_i, T_{i+1}) is

$$TC(k_i) = \int_{T_i}^{T_{i+1}} [(1 - \sum_{x=1}^{k_i} P(l_{i,x}, t)) + \lambda \times k_i] dt \quad (9)$$

We use the following algorithm to minimize (**10**).

$k_i = 1;$
while $(TC(k_i) > TC(k_i + 1))$
$k_i = k_i + 1;$

We then calculate the best initial paging area $PA_0(T_i)$ for time region (T_i, T_{i+1}) and we denote the corresponding location tracking cost as $LC(T_i, T_{i+1})$ using the algorithm mentioned above.

4.3. Determining The Check Points of A Moving Period

The final problem we need to solve for BBS is how to partition the moving period of each mobile into appropriate time regions. To find the best partition for each mobile, the partition problem is formulated in (10).

$$\text{Min } \alpha \sum_{i=0}^{k-1} LC(T_i, T_{i+1}) + \beta C_c k \quad (11)$$

Subject to

$\{[T_i, T_{i+1}) \mid 0 \le i \le k\}$ a partition of $[0, D)$

Where

1. α and β are weights of bandwidth cost and computational cost.
2. C_c is the unit computational cost.
3. $LC(T_i, T_{i+1}) = \min \int_{T_i}^{T_{i+1}} [(1 - \sum_{x=1}^{k_\alpha} P(l_{i,x}, t)) + \lambda \times k_i] dt$
4. $P(l, t)$ is the probability that the mobile stays inside l at time t.
5. $\bigcup_{j=1}^{k_i} \{l_{i,j}\} = PA_0(T_i)$
6. λ is the call arrival rate of the mobile.

To solve this problem, a greedy algorithm is presented below.

Step 1: S = {[0, D)} and S' = {[0, D)} ;
Step 2: For each element (T, T')
If $\alpha LC(T, T') > \alpha(LC(T, (T+T')/2) + LC((T+T')/2, T')) + \beta C_c$
then $S' = S' \cup \{(T, (T+T')/2), ((T+T')/2, T')\} - \{(T, T')\};$
Step 3: If $S \neq S'$ *then* $S = S'$ *and goto Step 2.*
Step 4: End this algorithm.

5. Experimental Results

5.1 Simulation Design

In this section, we evaluate of the fixed paging area strategy (FS), time-based strategy (TBS) and behavior-based strategy (BBS). In our simulation, the radius of

each cell is 250 meters, A RA contains 9 base stations and a MSC contains 25 RAs. For TBS, a location update is hold every 5 minutes and the size of each paging area is calculated according to the formulations estimated in [10] with confidence $\gamma = 0.9$.

To simulate the moving of a mobile, the moving schedule be assigned to our simulator. A moving schedule of a mobile is a lattice in which each vertex has three attributes: the location, the expectative arrive time and expectative leave time. The exact leave time is decided according to the following rules: If the exactly arrive time is earlier than the expectative leave time, than the leave time is in normal distribution with mean = expectative leave time and variance = 1. The time unit is minute. If the exactly arrive time is latter than the expectative leave time, than the mobile will leave this vertex immediately when it arrive this vertex.

Each edge of a moving schedule has 9 attributes that are given as follows. The start and end vertex of the path, the width of the path, the maximum, average and minimum moving speed, the diffusion of the moving speed, the probability that the mobile will choose this path when it want to leave its start vertex and whether the moving is returnable or not. A mobile moves on each path is in Brownian motion according to the rules defined in the path.

5.2 Experimental Results

We make experiments to compare the three strategies by taking into account their update frequency distributions, paging area size distribution and radio-link cost versus call arrival rate. Fig. 4 below shows the distribution of location update frequencies of the three strategies of 9 days' statistics. Because the mobile is in high speed moving in the two time regions (6:16Am, 8:25AM) and (17:02PM, 19:20PM), the update frequency of fixed paging area strategy in the two regions are relatively higher than other time regions. Because the mobile object has high regular moving behaviors in the two regions, the BBS has high accuracy in predicting the paging area of the two time regions. The update frequency of TBS is constant with value 9. Because the mobile has low regular moving behaviors in the time region (12:00PM, 13:25PM), the mobile often moves outside the paging area the BBS estimated. In this time region the BBS's performance is worse than the other two strategies.

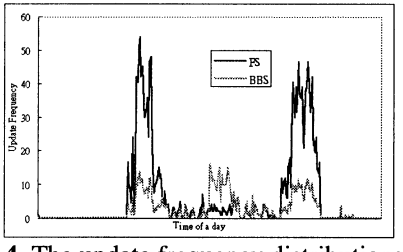


Fig. 4. The update frequency distributions.

Fig. 5 below shows the average size of paging area distributions of the two strategies BBS and TBS in the 9 days' statistic. The average paging area size of FS of all time regions are all 9 base stations (equal to 1 registration area). Because the mobile moves fast in the two regions (6:16AM, 8:25Am) and (17:02PM, 19:20PM),

the paging area size increases significantly in the two regions. Although the paging area size of the two strategies both depends on the moving speed of the mobile object, the performance of BBS is much better than TBS under high speed moving situation.

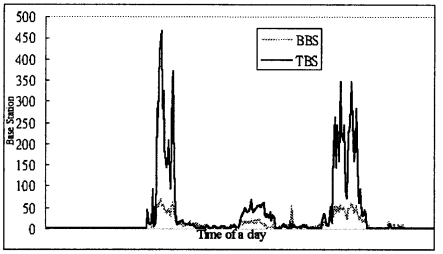


Fig. 5. The paging area size distributions

Fig. 6 below shows the radio link cost distributions of the three strategies under differ call arrival rate. The radio link cost is defined to be the sum of location update cost and paging cost presented in eq. 1 and eq. 2 respectively.

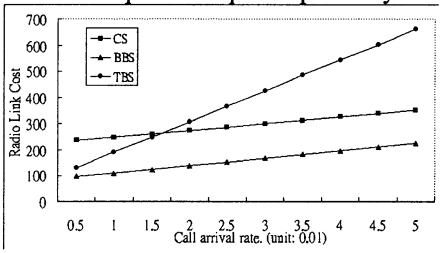


Fig. 6. Radio-link cost comparisons of the three strategies under different call arrival rates.

6. Conclusion

In this paper, we proposed a new version of profile-based strategy called behavior-based strategy. With the help of moving behavior, we estimate the time-varying probability and could estimate the expected cost for each time region whenever the paging area is defined for the time region. To calculate the time region, we construct a cost model to measure the cost for each assignment of checkpoints. The experimental results show that our strategy does save a great amount of radio link resource that is the most expensive cost for each location query.

Although the BBS has good performance in time region with regular moving behavior, future works include improving the performance for some time region with worse regularity. For example, in our simulation in section 5, although the moving in the time region (12:00PM, 13:25PM) is not regularity (it randomly walks in this time region), its moving speed in this region is not too fast and in most time it stays in the base stations closer to its office. If we use the registration area that contains the base station that covers the office of the mobile, then the update frequency will be close to FS. The performance would be improved furthermore. Channel assignment is another important issue for reducing the frequency of hand-off activity. To reduce the handoff

frequency, hierarchical cell structures seem to be a compromise between efficient uses of available channels while simultaneously keeping the number of handoffs small. Fast-moving mobiles are assigned to larger cells, and stationary or slow-moving mobiles are allocated to microcells. Now we are trying to mine the useful information from moving behavior for reducing the handoff frequency.

References

[1] I. F. Akyildiz and S. M. Ho, "On Location Management for Personal Communications Networks", IEEE Communications Magazine, Sep. 1996, pp. 138-145.

[2] S. Tabbane, ESPTT, "Location Management Methods for Third-Generation Mobile Systems", IEEE Communication Magazine, Aug. 1997, pp. 72-84.

[3] M. van Steen, F. J. Hauck, P. Homburg and A. S. Tanenbaum, "Locating Objects In Wide-Area Systems", IEEE Communication Magazine, pp.104 – 109, Jan. 1998.

[4] Y. B. Lin and S. K. Devries, "PCS Network Signaling Using SS7," IEEE Communication Magazine, pp.44-55, June 1995.

[5] T. P. Wang, S. Y. Hwang, and C. C. Tseng, "Registration Area Planning for PCS Networks Using Genetic Algorithms", IEEE Transactions on Vehicular Technology, vol. 47, No, 3, August 1998, pp 987-995.

[6] D. J. Goodman, H. Xie, "Intelligent Mobility Management for Personal Communications", IEE Colloquium on Mobility in Support of Personal Communications, London, England, June 16, 1993.

[7] H. Xie, S. Tabbane, D. J. Goodman, "Dynamic Location Area Management and Performance Analysis", Proc. 43rd IEEE VTC'93, Secaucus, NJ, May 1993, pp. 536-539.

[8] C. Rose and R. Yates, "Minimizing the Average Cost of Paging Under Delay Constraints," ACM-Baltzer J. Wireless Networks, vol. 1, no. 2, July 1995, pp. 211-19.

[9] C. Rose, "Minimizing the Average Cost of Paging and Registration: A Timer-Based Method", ACM/Batzer Journal of Wireless Networks, vol. 2, no. 2, pp. 109-116, 1996.

[10] Z. Lei, C. U. Saraydar and N. B. Mandayam, "Mobility Parameter Estimation for the Optimization of Personal Paging Areas in PCS/Cellular Mobile Networks", IEEE Workshop on Signal Processing Advances in Wireless Communications, May, 1999.

[11] U. Madhow, M. L. Honig, K. Steiglitz, "Optimization of Wireless Resources for Personal Communications Mobility Tracking," IEEE Transaction on Networking, Vol. 3, No, 6, pp. 698 – 707, 1995.

[12] A. Bar-Noy, I. Kessler and M. Sidi, "Mobile Users: To Update or Not to Update?", ACM/Batzer Journal for Wireless Networks, vol. 1, no. 2, July 1995, pp. 175-186.

[13] S. Tabbane, "An Alternative Strategy for Location Tracking", IEEE Journal on selected areas in communications, vol. 13, no. 5, June 1995

Extending RDBMS for Allowing Fuzzy Quantified Queries

Leonid José Tineo Rodríguez [1]

[1] Departamento de Computación, Universidad Simón Bolívar
Apartado 89000, Caracas 1080-A, Venezuela
leonid@ldc.usb.ve

Abstract. This paper is mainly concerned with the extension of database management systems querying capabilities, so that users may address queries involving preferences and get discriminated answers. The use of flexible predicates and linguistic quantifiers interpreted in the framework of the fuzzy set theory is advocated for defining a query language, called SQLf. This language extents the functionalities offered by SQL and it is considered here from a query processing point of view. We concentrate this work in the fuzzy quantified SQLf queries semantic and evaluation mechanism.

1 Introduction

It is often said that commercial DBMS suffer from a lack of flexibility [2],[8]; despite the tremendous evolution of this area in the last decade, imprecision has not been taken into account. In fact, a twofold hypothesis has been maintained: data are assumed to be precisely known and queries are intended to retrieve elements that qualify for a boolean condition. This paper concentrates on the second aspect of this hypothesis. The objective is to provide users with new flexible querying capabilities.

In the remainder of this paper, ordinary relational DBMS are considered. In this context, several approaches for the expression of preferences inside queries have been proposed. It has been shown in [1] that the solution founded on fuzzy sets is the most general one. An extension of SQL, called SQLf, supporting fuzzy queries in the context of a relational databases querying has been proposed [2]. Exist other extensions of SQL for flexible querying[10],[12],[14], but SQLf is the most complete and the only one that allows fuzzy quantified queries[11],[16]. We present here the semantic of such queries [3],[7].

An important point concerns the evaluation of fuzzy queries. For fuzzy queries the process becomes complex for two reasons: i) the access paths can not be directly used, and ii) a larger number of tuples is selected by fuzzy conditions with respect to boolean ones. So, it appears useful to understand the existing connections between properties tied to boolean conditions and fuzzy ones, so that fuzzy query processing can (partly) come down to Boolean query processing. An evaluation method, called derivation, exploiting such properties is described in [1],[4],[6]. The applicability of this method to fuzzy quantified queries is discussed here, as well as the integration of a derivation-based SQLf query interface on top of a regular relational DBMS.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 407-416, 2000.

2 Fuzzy Quantifiers

Quantifiers are used to represent the amount of items satisfying a condition. Classic logic has only two quantifiers: ∀ and ∃. In an attempt flexibling it, Zadeh[18] introduced the linguistic quantifiers. These quantifiers are represented as fuzzy sets, so they are called Fuzzy Quantifiers. Two types of fuzzy quantifiers are distinguished: The absolute that represent amounts that are absolute in nature (as "about 5", "more than 20"), they are represented by a fuzzy subset Q, such that for any nonnegative real p∈R+ the membership grade of p in Q ($\mu_Q(p)$) indicates the degree to which p is compatible with the quantifier; And the proportional quantifiers (as "at least half", "most"), represented by fuzzy subsets of the unit interval [0,1], for any prportion p∈[0,1], $\mu_Q(p)$ indicates the degree to which p is compatible with the meaning of the quantifier. Yager [17] has investigated a number of issues related to linguistic quantifiers. Dubois and Prade[10] have also contributed to the development of these objects. Functionally, linguistic quantifiers are usually of one of three types:

Increasing quantifiers (as "at least n", "all", "most") are characterized by:

$$\forall a \forall b \left((a < b) \rightarrow \left(\mu_Q(a) \leq \mu_Q(b) \right) \right) \tag{1}$$

Decreasing quantifiers (as "a few", "at most n") are characterized by:

$$\forall d \forall e \left((d < e) \rightarrow \left(\mu_Q(d) \geq \mu_Q(e) \right) \right) \tag{2}$$

Unimodal quantifiers (as "around n") have the property that:

$$\exists c \left(\left(\mu_Q(c) = 1 \right) \wedge \left(\begin{array}{l} \forall a \forall b \left((a < b \leq c) \rightarrow \left(\mu_Q(a) \leq \mu_Q(b) \right) \right) \wedge \\ \forall d \forall e \left((c \leq d < e) \rightarrow \left(\mu_Q(d) \geq \mu_Q(e) \right) \right) \end{array} \right) \right) \tag{3}$$

3 Fuzzy Quantifiers Interpretation

The interpretation of fuzzy quantifiers has received attention from several researchers [9],[13],[15],[17],[19]. But there is not one totally appropriate for database querying [11] and that can be evaluated in an efficient way[5],[6]. It would be of supreme interest to get such an interpretation. We present an interpretation that is a natural extension of quantifiers of classical logic, based in a linguistic transformation principle[16]. Due to the kind of fuzzy quantified queries to be studied in this work, we concentrate in quantified statements of the form "*Q X's ARE A*" (for short *Q(X,A)*), where *Q* is a fuzzy quantifier, *X* is a regular set and *A* is a fuzzy predicate.

If *Q* is increasing absolute the satisfaction degree of the sentence *Q(X,A)* is:

$$\mu\left(Q(X,A)\right) = \sup_{i \in \{0..|X|\}} \left(\min \left(\mu_Q(i), \sup_{x \in X} \left(\mu_A(x) \right) \right) \right) \tag{4}$$

If Q is decreasing absolute, the satisfaction degree of the sentence $Q(X,A)$ is:

$$\mu_{(Q(X,A))} = \sup_{i \in \{0..|X|\}} \left(\min \left(\mu_Q(i), \inf_{x \in X}{}^{i+1} \left(1 - \mu_A(x) \right) \right) \right) \quad (5)$$

If Q is unimodal absolute the satisfaction degree of the sentence $Q(X,A)$ is:

$$\mu_{(Q(X,A))} = \min \left(\begin{array}{l} \sup_{l \in \{0..|X|\}} \left(\min \left(\mu_Q(l), \sup_{x \in X}{}^{l} \left(\mu_A(x) \right) \right) \right), \\ \sup_{r \in \{0..|X|\}} \left(\min \left(\mu_Q(r), \inf_{x \in X}{}^{r+1} \left(1 - \mu_A(x) \right) \right) \right) \end{array} \right) \quad (6)$$

For proportional quantifiers the satisfaction degrees are similar to previous aones but changin the argumente of the quantifier membership fuction by the proportion of the index respect the cardinality of the set X.

Regarding with other interpretations for fuzzy quantifiers, our interpretation is comparable to that of Prade[13]. Our interpretation for sentences of the form "*Q X's ARE A*" is equivalent to the Possibility measure given by Prade's for increasing quantifiers. And in case of decreasing quantifiers, our interpretation for sentences of the form "*Q X's ARE A*" is equivalent to the Prade's Necessity measure. Prade does not deals with unimodal quantifiers. There is also a similarity between our interpretation and that of Ralescu[15] for sentences of the form "*Q X's ARE A*". But Ralescu gives an integer number for the cardinality and we give a fuzzy integer[16].

4 Fuzzy Quantified Queries in SQLf

SQL has been extended in a straightforward manner to allow flexible querying. This gave birth to SQLf [2]. SQLf has the same general SQL philosophy (querying features and syntax in particular) and offers new possibilities for flexible querying.

In this section, we present the single-block partitioned queries with fuzzy quantifier, which constitute the kind of queries that we will consider in the following from an evaluation point of view. The structure of such queries is: "SELECT t A FROM R GROUP BY A HAVING Q fc". where t is a threshold associated with the query (is optional), A is an attribute (or attribute list) of relations in R, Q is any fuzzy quantifier, fc is a fuzzy condition. This query returns the fuzzy relation R_f such that

$$dom(R_f) = \left\{ a \Big/ \left((\exists x \in R(x.A = a)) \wedge \left(\mu_{(Q(X_a, fc))} \geq t \right) \right) \right\} \quad (7)$$

$$\forall a \in dom(R_f) \left(\mu_{R_f}(a) = \mu_{(Q(X_a, fc))} \right) \quad (8)$$

Where

$$X_a = \{x \in R / x.A = a\} \quad \textbf{(9)}$$

Example 1: Semantic of a fuzzy quantified query

Let's consider the relation EMP in Table 1, the fuzzy quantifier "MostOf" and the fuzzy predicate "About40" definded by membership functions in Fig. 1.

Table 1. EMP relation extension

#emp	e-name	Salary	Job	Age	#dep
10	Martin	2000	K1	40	1
22	Calvin	1000	K4	38	1
78	Luther	1500	K2	50	1
41	Johnson	1200	K3	40	2
35	Smith	1000	K3	39	2
90	Peters	1200	K2	41	2
56	Anderson	1500	K2	40	3
82	Dobson	1000	K4	36	3
64	Mc Dowell	2000	K1	50	3

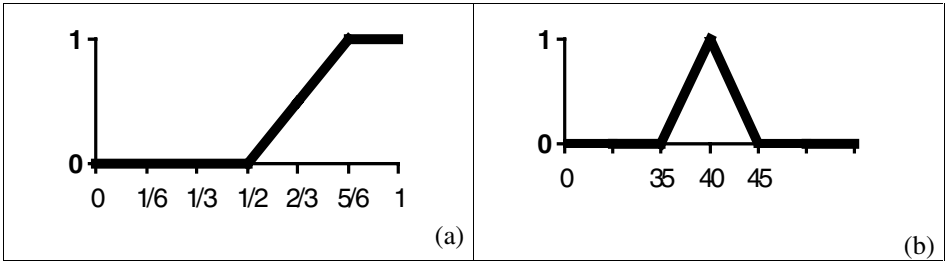


Fig. 1. (a) The quantifier "MostOf" and (b) The predicate "About40"

The query "Find the departments where most of the employees are about 40 years", with a threshold 0.5, may be expressed in SQLf by: "SELECT 0.5 #dep FROM EMP GROUP BY #dep HAVING MostOf age = About40". The fuzzy degree calculation and the resulting relation are in Table 2 and Table 3.

Table 2. Computation of satisfaction degrees for a query

#dep (*a*)	*I*	#emp	Age	$\mu_i : {}^{i}\sup_{x \in X_a} (\mu_{fc}(x))$	$\mu_{Qi} : \mu_Q\left(\frac{i}{\lvert X_a \rvert}\right)$	$\min(\mu_i, \mu_{Q_i})$	$\mu(Q(X_a, fc))$
1	1	10	40	1	0	0	.5
	2	22	38	.6	.5	.5	
	3	78	50	0	1	0	

2	1	41	40	1	0	0	.8
	2	35	39	.8	.5	.5	
	3	90	41	.8	1	.8	
3	1	56	40	1	0	0	.2
	2	82	36	.2	.5	.2	
	3	64	50	0	1	0	

Table 3. Fuzzy Query Result

#dep	Membership degree
1	.5
2	.8

5 Derivation Principle for SQLf Quantified Queries

An important issue concerns the processing of fuzzy queries. The strategy presented hereafter assumes that a threshold t is associated with a SQLf query in order to retrieve only those tuples that satisfy the condition with a degree greater or equal to t (the t-cut). The idea advocated here [6] is to use an existing database management system that will process boolean queries. An SQL query is derived from the SQLf expression in order to retrieve the t-cut. Then, the fuzzy query can be processed on this set avoiding the exhaustive scan of the database. Remark that the derived SQL query is not equivlent to the original SQLf. The SQL query delivers a regular set, while the result of the SQLf query is a fuzzy set. The principle is to express the t-cut with a query involving only crisp expressions. If the condition involves means operators as connectors, the derived condition produces a superset of the t-cut, in this case it is said that the derivation is weak, otherwise is said that it is strong.. The efficiency of the transformation in simple queries has been studied in [4]. The application of this principle to simple and compound predicates (without quantifiers) has been object of previous works [3,6]. Here we concentrate on the application of this principle to quantified queries.

Definition 1: Derived query for a simple fuzzy query.

Let's "SELECT t A FROM R WHERE fc" be a simple fuzzy query in SQLf. The **derived SQL query** for this query is: "SELECT A FROM R WHERE DNC(fc, ≥,t)". Where DNC(Q fc, ≥,t) denotes the **derived necessary condition** obtained from the initial WHERE condition "fc".

Definition 2: Derived query for a fuzzy quantified partitioned query.

Let's "SELECT t A FROM R GROUP BY A HAVING Q fc" be a fuzzy quantified query in SQLf. The **derived SQL query** for this query is: "SELECT A FROM R WHERE DNWC(Q fc, ≥,t) GROUP BY A HAVING DNHC(Q fc, ≥,t).". Where

DNWC(Q fc, ≥,t) denotes the **derived necessary condition for the WHERE clause** and DNHC(Q fc, ≥,t) **denotes the derived necessary condition for the HAVING clause** obtained from the initial HAVING condition "Q fc".

We demonstrate the applicability of the derivation principle here for the case of increasing absolute quantifiers. The demonstration for other quantifiers is similar, we don't present them here due to space reasons, see [16] for details.

Proposition 1

Let's Q be a fuzzy quantifier, fc a fuzzy condition, "SELECT t A FROM R R_l GROUP BY A HAVING Q fc" a fuzzy quantified query (R_l is an alias for R).
If Q is Absolute: Let's l be the minimum value such that $\mu_Q(l)\geq t$, r be the maximum value such that $\mu_Q(r)\geq t$
If Q is Increasing Absolute then:

$$((\text{DNWC}(\text{Q fc},\geq,t)=\text{DNC}(\text{fc},\geq,t))\wedge(\text{DNHC}(\text{Q fc},\geq,t)=(\text{count}(*)\geq 1))) \quad \textbf{(10)}$$

If Q is Decreasing Absolute then:

$$\begin{pmatrix}(\text{DNWC}(\text{Q fc},\geq,t)=\text{DNC}(\text{fc},\leq,1-t))\wedge\\(\text{DNHC}(\text{Q fc},\geq,t)=(\text{count}(*)+r\geq(\text{SELECT count} * \text{FROM R WHERE A = R1.A})))\end{pmatrix} \quad \textbf{(11)}$$

If Q is Unimodal Absolute then:

$$\left(\begin{pmatrix}(\text{DNWC}(\text{Q fc},\geq,t)=TRUE)\wedge\\\left(\begin{pmatrix}\text{DNHC}(\text{Q fc},\geq,t)=\\\left(\begin{pmatrix}(\text{l}\leq\text{SELECT count} * \text{FROM R WHERE A = R1.A AND DNC}(\text{fc},\geq,\text{t}))AND\\(\text{r}\geq\text{SELECT count} * \text{FROM R WHERE A = R1.A AND DSC}(\text{fc},>,1-\text{t}))\end{pmatrix}\right)\end{pmatrix}\right)\end{pmatrix}\right) \quad \textbf{(12)}$$

If Q is Proportional: Let's q_l be the minimum value such that $\mu_Q(q_l)\geq t$, q_r be the maximum value such that $\mu_Q(q_r)\geq t$
If Q is Increasing Proportional then:

$$\begin{pmatrix}(\text{DNWC}(\text{Q fc},\geq,t)=\text{DNC}(\text{fc},\geq,t))\wedge\\(\text{DNHC}(\text{Q fc},\geq,t)=(\text{count}(*)/\,ql\geq(\text{SELECT count} * \text{FROM R WHERE A = R1.A})))\end{pmatrix} \quad \textbf{(13)}$$

If Q is Decreasing Proportional then:

$$\begin{pmatrix}(\text{DNWC}(\text{Q fc},\geq,t)=\text{DNC}(\text{fc},\leq,1-t))\wedge\\(\text{DNHC}(\text{Q fc},\geq,t)=((\text{count}(*)/(1-q_r))\geq(\text{SELECT count} * \text{FROM R WHERE A = R1.A})))\end{pmatrix} \quad \textbf{(14)}$$

If Q is Unimodal Proportional then:

$$\left(\begin{pmatrix}(\text{DNWC}(\text{Q fc},\geq,t)=TRUE)\wedge\\\left(\begin{pmatrix}\text{DNHC}(\text{Q fc},\geq,t)=\\\left(\begin{pmatrix}(\text{count}(*)*q_l\leq\text{SELECT count} * \text{FROM R WHERE A = R1.A AND DNC}(\text{fc},\geq,\text{t}))\\AND\\(\text{count}(*)*q_r\geq\text{SELECT count} * \text{FROM R WHERE A = R1.A AND DSC}(\text{fc},>,1-\text{t}))\end{pmatrix}\right)\end{pmatrix}\right)\end{pmatrix}\right) \quad \textbf{(15)}$$

Proof (of 10)
Let's Xa be as in (9). By (4) we have

$$\left(\mu_{(Q(X_a, fc))} \geq t\right) \Leftrightarrow \left(\sup_{i \in \{0..|X_a|\}} \left(\min\left(\mu_Q(i), {}^{i}\sup_{x \in X_a}\left(\mu_{fc}(x)\right)\right)\right) \geq t\right) \tag{16}$$

Let's n be the lowest natural number such that $\left(\mu_Q(n) \geq t\right)$ then by (1):

$$\forall i \in \{0..n-1\}\left(\min\left(\mu_Q(i), {}^{i}\sup_{x \in X_a}\left(\mu_{fc}(x)\right)\right) < t\right) \tag{17}$$

and

$$\forall i \in \{n..|X_a|\}\left(\left(\min\left(\mu_Q(i), {}^{i}\sup_{x \in X_a}\left(\mu_{fc}(x)\right)\right) \geq t\right) \Leftrightarrow \left({}^{i}\sup_{x \in X_a}\left(\mu_{fc}(x)\right) \geq t\right)\right) \tag{18}$$

Combining (9),(16),(17) and (18), we may obtain:

$$\left(\mu_{(Q(X_a, fc))} \geq t\right) \Leftrightarrow \left(\left|\left\{x \in R \middle/ \left(\left(\mu_{fc}(x) \geq t\right) \wedge \left(X.A = a\right)\right)\right\}\right| \geq n\right) \tag{19}$$

Finally (10) follows of rewriting (19) in SQL sintax.

Example 2: Query Derivation

Find "The departments where at least 10 employees are about 40 years", with a threshold 0.5, "About40" defined in Fig. 1, and "AtLeast10" defined in Fig. 2. We write this query in SQLf as: "SELECT 0.5 #dep FROM Emp GROUP BY #dep HAVING AtLeast10 (age=About40)."

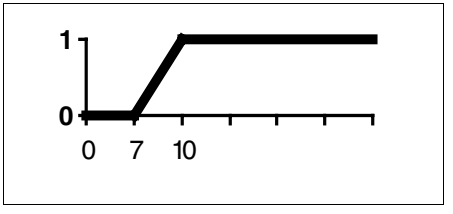


Fig. 2. Representation of the quantifier "AtLeast10"

An age a satisfy About40 with a threshold 0.5 iff 37.5≤a≤42.5. And The lowest n such that AtLeast10(n) ≥0.5 is n=9. Therefore: DNWC(AtLeast10 (age=About40), ≥,0.5) = (37.5≤age AND age≤42.5); and DNHC(AtLeast (age=About40), ≥,0.5) = count(*)≥9. Finally, the derived SQL query is: "SELECT #dep FROM Emp WHERE 37.5<=age AND age<=42.5 GROUP BY #dep HAVING count(*)≥9."

6 Evaluation of Fuzzy Quantified Queries on Top of a RDBMS

We present an evaluation mechanism based in the derivation principle. It is then possible to use an existing RDBMS to process the derived query. In so doing, one can expect to take advantage of the implementation mechanisms handled by the RDBMS to reach acceptable performances. The architecture for this strategy is shown in Fig. 3. The translation mechanism generates derived SQL statements that are processed by a procedural evaluation program. This architecture ensure a reduced development effort.

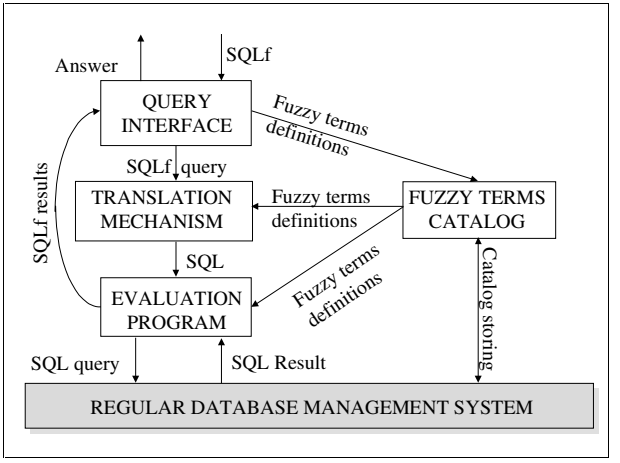


Fig. 3. Architecrure for implementing SQLf

For computing the membership of each element, one must execute an algorithm that processes the result of the derived query. We give here the external program, written in a host language, for partitioned queries with increasing absolute fuzzy quantifier.

Program 1**:** Evaluation of Partitioned Query with Increasing Absolute Quantifier

```
declare c cursor for      "SELECT * FROM R WHERE DNC(fc,>=,t)
                           GROUP BY A HAVING count(*)>=n."
result:=empty;
open c;
fetch c into x;
while code(c)<> Active Set Empty do
    A:=x.A;
    sup[0]:= 1;        (* At Least None <=> True *)
    sup[1]:= 0;        (* End mark for decreasing list*)
        while code(c)<>ActiveSetEmpty and x.A=A do ( *Grouping *)
        μ := μfc(x);   (* satisfaction degree of fc(x) *)
        if (μ>=t) then    (* calibration *)
            DecInsert(μ,sup); (* sup: μfc(x)degrees *)
                                              (* decreasing list*)
        fetch c into x;
    end while;
    Sup := 0;
```

```
    i := n;
    while (sup[i]<>0) do (* Cuantif.Stat. degree computation*)
        μ := min(μQ(i),sup[i]);
        if (μ>Sup) then Sup:=μ;
        i:=i+1;
    end while;
    result:=result+{Sup/A}   (* A degree is Sup *)
end while;
close c;
end program.
```

The previous program delivers the SQLf result without order. However one should want to see the result in satisfaction degree decreasing order of tuples. It can be performed changing in the propram the set "result" by a list and making oredered insertions, or viewing the result with and ordering filter. We have developed the algorithms for evaluating queries with all kind of quantifiers via the derivation principle. They are very similar to the previous one. We do not present them here by space restriction, see [16] for details.

About the Efficiency of the Derivation Principle: The derivation principle for quantifiers is strong, we have shown that via equivalences. Only when the derivation of the condition "fc" under the quantifier is weak, the derived query delivers extra tuples, it occurs only if "fc" contains means operators. The proportion of additional tuples in this case (seen as an efficiency index) is examined in [4]. The use of the derivation principle for the evaluation of the query in the example 1 would avoid the degrees calculation for the department 3 and for the employee 78 (4 tuples of 9). It also would avoid the computation membership degrees and minimum for portions 1/3 (2 of 5 calculations). We have developed a prototype of SQLf based on the derivation and another naive, with them we are carrying out tests of efficiency.

7 Conclusion

We have dealt with RDBMS with conventional data, which support fuzzy quantified queries. The semantics and processing of such queries have been discussed. We have presented an interpretation of fuzzy quantifiers, wich is completely suitable for database querying, it satisfies all Lietard's postulates [11] (other interpretations do not satisfy all of the postulates, only Yager's OWA interpretation [17] satisfies them also). It has been shown that, with this interpretation, it is possible to derive boolean queries, which return a t-cut of the initial fuzzy query (it was not possible other interpretations.). Then, the fuzzy query can be processed on this set avoiding the the whole database exhaustive scan. The major interest of this approach is to take advantage of the implementation techniques available in existing RDBMS. Other extensions of RDBMS have been made to allow flexible querying, but none of them allows quantified queries. We are making experimentation with a prototype of SQLf for showing the efficiency of the presented mechanism. In SQLf is also possible to use the fuzzy quantifiers as nesting operators, this kind of queries may also be evaluated via the derivation, but it is matter of future research.

References

1. P. Bosc, O. Pivert “Some approaches for relational databases flexible querying”, International Journal of Intelligent Systems, Vol 1, No. 3/4, pp. 323-354, Feb 1992.
2. P. Bosc, O. Pivert “SQLf: A Relational Database Language for Fuzzy Querying”, IEEE Transactions on Fuzzy Systems, Vol 3, No. 1, Feb 1995.
3. P. Bosc, O. Pivert, K.Farquhar “Integrating Fuzzy Queries into an Existing Database Management System: An Example ”, Intern. Journal of Intelligent Syst., Vol 9, pp 475-492,1994
4. P. Bosc, O. Pivert “On the efficiency of the alpha-cut distribution method to evaluate simple fuzzy relational queries ” , Advances in Fuzzy Systems-Applications and Theory, Vol 4, Fuzzy Logic and Soft Computing, B. Bouchon-Meunier, R.R.Yager,L.A. Zadeh eds, Wold Scientific, pp 251-260, 1995.
5. P. Bosc, L. Liétard, O. Pivert, “Quantified statements and Database Fuzzy querying”, Fuzziness in Database Management Systems, P. Bosc & J. Kacprzyk Eds, Physica Verlag, pp. 275-308,1995.
6. P. Bosc, O. Pivert, "SQLf query functionality on top of a regular relational DBMS", in: Knowledge Management in Fuzzy Databases, O. Pons, M.A. Vila, and J. Kacprzyk (Eds.), Heidelberg: Physica-Verlag, to appear.
7. Buckles, J.Buckley & F.Petry “Architecture of FAME: Fuzzy Address Mapping Environment”, Proc. of 3rd IEEE Intern. Conference on Fuzzy Syst., pp 308-312, 1994.
8. E. Cox “Relational Database Queries using Fuzzy Logic”, Artificial Intelligent Expert, pp 23-29, Jan 1995.
9. Dubois, H. Prade “Fuzzy cardinality and the modeling of imp recise quantification, Fuzzy Sets and Systems”, Vol. 16, No. 2, pp 199-230, 1985.
10. J.Kacpryzyk & S.Zadrozny, “Fuzzy Queries in Microsoft Access™v.2”, Proceedings of Fuzzy IEEE Workshop on Fuzzy Database Syst. and Information Retrieval, pp 61-66, 1995.
11. L. Lietard “Contribution `a Línterrogation Flexible de Bases de Données: étude des propositions quantifiées floues” These de Docteur de Lúniversité de Rennes
12. H.Nakajima, T.Sogoh, M.Arao “Fuzzy Database Language and Library-Fuzzy Extension to SQL”, Proceedings of 2nd IEEE Int. Conference on Fuzzy Syst., pp 477-482, 1983.
13. H. Prade, “A tow-layer fuzzy pattern matching procedure for the evaluation of conditions involving vague quantifiers”, Journal of Intelligent and Robotics Syst., No. 3, pp 93-101, 1990.
14. F. E. Petry “Fuzzy Databases Principles and Applications” International Series in Intelligent Technologies, Klumer Academic Publishers.
15. D. Ralescu "Cardinality, quantifiers, and the aggregation of fuzzy criteria" Fuzzy Sets and Systems 69, pp 355-365. 1995
16. L. Tineo, “Interrogaciones Flexibles en Bases de Datos Relacionales”, Universidad Simón Bolívar, Technical Report, Jan 1998.
17. R. Yager, "Connectives and quantifiers in fuzzay sets" Fuzzy Sets Syst. 40, 39-76 (1991).
18. L.A. Zadeh, "A theory of approximate reasoning" in Machine Intelligence, Vol. 9, J. Hayes, D. Michie, and L. I. Mikulich, Eds., Halstead Press, New York, 1979, pp. 149-194.
19. L.A. Zadeh ,”A computational approch to fuzzy quantifiers in natural languages”, Computer Mathematics with Applications, No. 9, pp 149-183, 1983.

Knowledge Decay in a Normalised Knowledge Base

John Debenham

University of Technology, Sydney,
School of Computing Sciences,
PO Box 123, NSW 2007, Australia
debenham@socs.uts.edu.au

Abstract. Knowledge 'decay' is a measure of the degradation of knowledge integrity. In a unified knowledge representation, data, information and knowledge are all represented in a single formalism as "items". A unified knowledge representation is extended here to include two measures of knowledge integrity. A rule of inference is defined on the unified knowledge representation that preserves the validity of these two measures. This rule of inference is used to define a normalised knowledge base. The use of a unified knowledge representation and the application of knowledge base normalisation simplifies the estimation of knowledge decay.

1. Introduction

Knowledge 'decay' is a measure of the degradation of knowledge integrity. In a *unified* knowledge representation, data, information and knowledge are all represented in a single formalism as "items". A unified model is expressed in terms of "items" and "objects" [1]; objects are item-building operators. A rule of inference, called item (and object) "join" is defined [2]. That rule is extended here so that it accommodates the two measures of knowledge integrity. A single rule for "knowledge decomposition" is defined in terms of that rule. Knowledge decomposition is applied to normalise the knowledge in a knowledge base. Classical database normalisation [3] is a special case of knowledge base normalisation. The use of a unified knowledge representation and the application of knowledge base normalisation simplifies the estimation of knowledge decay.

2. Unified knowledge representation

The terms 'data', 'information' and 'knowledge' are used here in a rather idiosyncratic sense [1]. The *data* in an application are those things that can be represented as simple constants or variables. The *information* is those things that can be represented as tuples or relations. The *knowledge* is those things that can be represented either as programs in an imperative language or as rules in a declarative language. Items are a unified knowledge representation; they have a uniform format no matter whether they represent 'data', 'information' or 'knowledge' things.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 417–426, 2000.

The key to the unified representation is the way that the meaning of an item, the item 'semantics' is specified. The *semantics* of an item is a function that *recognises* the members of the "value set" of that item. The *value set* of an information item is the set of tuples that are associated with a relational implementation of that item. Knowledge items, including complex, recursive knowledge items, have value sets too [1]. For example, the item, which represents the rule "the sale price of parts is the cost price marked up by a universal mark-up factor", could have a value set of quintuples associated with *[part/sale-price, part/cost-price, mark-up]*.

The notion of an item is extended here to incorporate two integrity measures. The *item tuple integrity measure* for an item is a measure of the likelihood that a given tuple should *not* belong to the value set of that item. The *item set integrity measure* for an item is a measure of the likelihood that a given set is an invalid value set of that item. Items are *either* represented informally as "i-schema" *or* formally as λ-calculus expressions. The i-schema notation is used in applications. Formally, given a unique name A, an n-tuple $(m_1, m_2,..., m_n)$, $M = \Sigma_i\ m_i$, if:

- S_A is an M-argument expression of the form:
  $\lambda y^1_1...y^1_{m_1}...y^n_{m_n} \bullet [S_{A_1}(y^1_1,...,y^1_{m_1}) \wedge .. \wedge S_{A_n}(y^n_1,...,y^n_{m_n}) \wedge J(y^1_1,..,y^1_{m_1},..,y^n_{m_n})] \bullet$
  where $\{A_1,..., A_n\}$ is an ordered set of not necessarily distinct items, each item in this set is called a *component* of item A.
- V_A is an M-argument fuzzy expression of the form:
  $\lambda y^1_1...y^1_{m_1}...y^n_{m_n} \bullet [V_{A_1}(y^1_1,...,y^1_{m_1}) \wedge .. \wedge V_{A_n}(y^n_1,..,y^n_{m_n}) \wedge K(y^1_1,..,y^1_{m_1},..,y^n_{m_n})] \bullet$
  where $\{A_1,..., A_n\}$ are the components of item A, K is a fuzzy predicate and $\wedge$ is the fuzzy "min" conjunction.
- C_A is a fuzzy expression of the form:
  $[\ C_{A_1} \wedge C_{A_2} \wedge ... \wedge C_{A_n} \wedge (L)_A\ ]$
  where $\wedge$ is the fuzzy "min" conjunction and L is a fuzzy expression constructed as a logical combination of:
  - Card_A lies in some numerical range;
  - $\text{Uni}(A_i)$ for some i, $1 \le i \le n$, and
  - $\text{Can}(A_i, X)$ for some i, $1 \le i \le n$, where X is a non-empty subset of $\{A_1,..., A_n\} - \{A_i\}$;

  subscripted with the name of the item A,

then the named triple $A[\ S_A, V_A, C_A]$ is an M-adic *item* with *item name* A, S_A is called the *item semantics* of A, V_A is called the *item tuple integrity measure* of A and C_A is called the *item set integrity measure* of A. "$\text{Uni}(A_i)$" is a fuzzy predicate whose truth value is "the proportion of the members of the value set of item A_i that also occur in the value set of item A". "$\text{Can}(A_i, X)$" is a fuzzy predicate whose truth value is "in the value set of item A, the proportion of members of the value set of the set of items X that functionally determine members of the value set of item A_i,". "Card_A" means "the number of different values in the value set of item A". The

subscripts identify the item's components to which that measure applies. Given an item A and tuple X, if $V_A(X) = 1$ then this does *not* imply that X is invalid in the value set of A; if $V_A(X) = 0$ then X is invalid in the value set of A. Likewise for C_A. The measures are *not* measures of validity. A value of unity does *not* imply validity. An item's semantics specifies what *should be* in an implementation, and the two integrity measures are measures of invalidity of what *is* in an implementation.

For example, an application may contain an association whereby each *cost-price* is associated with a *sales-tax* amount as determined by the prevailing *tax-rate*. This association could be represented by the knowledge item:

$$
\begin{aligned}
&\textit{[cost-price, sales-tax, tax-rate]}[\\
&\quad \lambda xyz\bullet[\, S_{\textit{cost-price}}(x) \wedge S_{\textit{sales-tax}}(y) \wedge S_{\textit{tax-rate}}(z) \wedge (y = x \times z)]\bullet,\\
&\quad \lambda xyz\bullet[\, V_{\textit{cost-price}}(x) \wedge V_{\textit{sales-tax}}(y) \wedge V_{\textit{tax-rate}}(z) \wedge K_2(x, y, z)\,]\bullet\\
&\quad [\, C_{\textit{cost-price}} \wedge C_{\textit{sales-tax}} \wedge C_{\textit{tax-rate}} \wedge\\
&\qquad (\,\mathrm{Can}(\textit{cost-price}, \{\textit{sales-tax}, \textit{tax-rate}\}) \wedge\\
&\qquad\quad \mathrm{Can}(\textit{sales-tax}, \{\textit{cost-price}, \textit{tax-rate}\}) \wedge\\
&\qquad\quad \mathrm{Can}(\textit{tax-rate}, \{\textit{cost-price}, \textit{sales-tax}\})\,)_{\textit{[cost-price, sales-tax, tax-rate]}}\,]\,]
\end{aligned}
$$

where:

$$
K_2(x, y, z) = \begin{cases} 0 & \textit{if } x \neq 0 \textit{ and } y \neq 0 \\ \dfrac{x}{x + 100 \times |\, z \times x - y \,|} & \textit{otherwise} \end{cases}
$$

Items make it difficult to analyse the structure of the whole application because, for example, two rules that share the same basic wisdom may be expressed in terms of quite different components; this could obscure their common wisdom. To make the inherent structure of knowledge clear 'objects' are introduced as item building operators [4].

Object names are written in bold italics. Suppose that the conceptual model already contains the item "*part*" which represents spare parts, and the item "*cost-price*" which represents cost prices; then the information "spare parts have a cost price" can be represented by "*part/cost-price*" which may be built by applying the "***costs***" object to *part* and *cost-price*:

part/cost-price = ***costs***(*part*, *cost-price*)

Suppose that the conceptual model already contains the item "*part/sale-price*" which represents the association between spare parts and their corresponding selling price, and the item "*mark-up*" which represents the data thing a universal mark-up factor; then the rule "spare parts are marked up by a universal mark up factor" can be represented by *[part/sale-price, part/cost-price, mark-up]* which is built by applying the "***mark-up-rule***" object to the items "*part/sale-price*", "*part/cost-price*" and "*mark-up*":

[part/sale-price, part/cost-price, mark-up] =
mark-up-rule(*part/sale-price*, *part/cost-price*, *mark-up*)

The conceptual model contains items. A fundamental set of data items in the conceptual model is called the *basis*. The remaining items in the conceptual model are built by applying object operators to the other items in the conceptual model. As

for items, objects may either be represented informally as "o-schema" or formally as typed λ-calculus expressions.

In [2] the composition of items and objects using a "join" operator is described. Join is extended here to include the measures of integrity. Decomposition is defined in terms of join. Knowledge base normalisation is defined in terms of decomposition [5]. Normalisation removes hidden relationships from the conceptual model. Hidden relationships can present a maintenance hazard [6]. Knowledge base normalisation simplifies the estimation of knowledge decay [7].

Item join provides the basis for item decomposition. The definition of "item composition" follows. Given two items:

$A[\ S_A,\ V_A,\ C_A\ ]$ and
$B[\ S_B,\ V_B,\ C_B\ ]$

Suppose that S_A has n variables, that is A is an n-adic item. Suppose that S_B has m variables, that is B is an m-adic item. Some of the components of A and B may be identical. Suppose that k pairs of components of A and B that are identical are identified, where $k \geq 0$. Let E be an ordered set of components where each is one of these identical pairs of components of both A and B. E may be empty. To ensure that the definition is well defined the order of the components in the set E is the same as order in which they occur as components of A. Suppose the semantics expressions of the components from item A (or item B) that are in the set E are expressed in terms of a total of p variables. Let A^* be an n-adic item that is identical to item A except for the order of its variables. The last p variables in A^* are those variables in A that belong to the components of A in the set E. Let B^* be an m-adic item that is identical to item B except for the order of its variables. The first p variables in B^* are those variables in B that belong to the components of B in the set E. Let π be a permutation that turns the ordered set of variables of A^* into the ordered set of variables of A. Let π' be a permutation that turns the ordered set of variables of B^* into the ordered set of variables of B. Suppose that 'x' is an (n - p)-tuple of free variables, 'y' is a p-tuple of free variables and 'z' is an (m - p)-tuple of free variables. Then the item with name $A \otimes_E B$ is the *composition* of A and B on E and is defined to be the item:

$(A \otimes_E B)[\ \lambda xyz\bullet[S_A(\pi(x,y)) \wedge S_B(\pi'(y,z))]\bullet,$
$\lambda xyz\bullet[V_A(\pi(x,y)) \wedge V_B(\pi'(y,z))]\bullet,\ C_{A \otimes_E B}\ ]$

where $C_{A \otimes_E B}$ is defined as follows. Suppose that C_A is an expression of the form $c_A \wedge G$ where c is that part of C_A that carries the subscript 'A' where any occurrence of the predicate Card are equated to "true", and G is that part of C_A that carries subscripts other than 'A'. Likewise suppose that C_B is an expression of the form $d_B \wedge H$. Then:

$C_{A \otimes_E B} = (c \wedge d)_{A \otimes_E B} \wedge (G \wedge H)$

The set E is *a* set of identical pairs of components of A and B. If E is the set of *all* identical pairs of components of A and B then $A \otimes_E B$ is written as $A \otimes B$.

If A and B are two information items, and if set E contains one component then $A \otimes_E B$ generalises the "join", in the conventional database sense, of the

relations related to A and B on the shared domain in the set E. Here this join contains the two classes of integrity estimates. Also, if A and B are two functional associations and if E contains a component that is both the domain of one of these functions and the range of the other then $A \otimes_E B$ is the functional composition of the two functions. The set E may be empty. If $E = \emptyset$ then $A \otimes_E B$ is the Cartesian product of A and B.

Given two n-adic items A and B with the same set of components that are not necessarily in the same order, A and B are *weakly equivalent*, written $A \simeq_W B$, if there exists a permutation π such that:

$(\forall x_1 x_2 ... x_n)[S_A(x_1,x_2,...,x_n) \Leftrightarrow S_B(\pi(x_1,x_2,...,x_n))]$

where the x_i are the n_i variables associated with the i'th component of A. The composition $A \otimes_E B$ is a *monotonic composition* if $A \otimes_E B$ is *not* weakly equivalent with either A or B.

Using the rule of composition $\otimes$, knowledge items, information items and data items may be joined with one another regardless of type. In this way items may be joined together to form more complex items. Alternatively, the $\otimes$ operator may form the basis of decomposition in which each item may be replaced by a set of simpler items. An item I is *decomposable* into the set of items $D = \{I_1, I_2,..., I_n\}$ if: I_i has non-trivial semantics for all i, $I = I_1 \otimes I_2 \otimes ... \otimes I_n$, where each composition is *monotonic*. If item I is decomposable then it will not necessarily have a unique decomposition [4]. A knowledge base is *normal* if it contains no decomposable items [1].

3. Knowledge Base Implementation

The semantics of an item $A[S_A, V_A, C_A]$ is a function that recognises the members of its value set. The value set is a conceptual notion in the system design. So the value set of the item A—as in the definition of an M-adic item above—at time τ is:

$$\gamma^{\tau}(A) = \{ y^1_1 ... y^1_{m_1} ... y^n_{m_n} : S_A(y^1_1,..,y^1_{m_1},...,y^n_{m_n}) \text{ at time } \tau \}$$

A *knowledge base implementation* is a set of knowledge items and a set of stored relations and data domains representing some information and data items. Some information and data items are associated with actual stored data, and some are not. Knowledge items are not normally associated with actual stored data. If an item is associated with actual stored data then it is a *real item*; otherwise it is a *virtual item*. The set of tuples in the implementation of the *real* item A is denoted by $\lambda^{\alpha}(A)$ where α is the time of the most recent modification to those tuples. Knowledge items may be used to derive tuples for virtual data and information items. For example, suppose that the real data item *mark-up* has a stored data value mark-up, and that the real information item *part/cost-price* has a stored relation part/cost-price. Then the knowledge item *[part/sale-price, part/cost-price, mark-up]*—or an "if-then" implementation of it—may be used to *derive* tuples in the relation for the virtual item *part/sale-price*. Further, the knowledge item *[part/sale-price, part/tax-payable, tax-rate]* could then enable the tuples in the relation for the virtual item

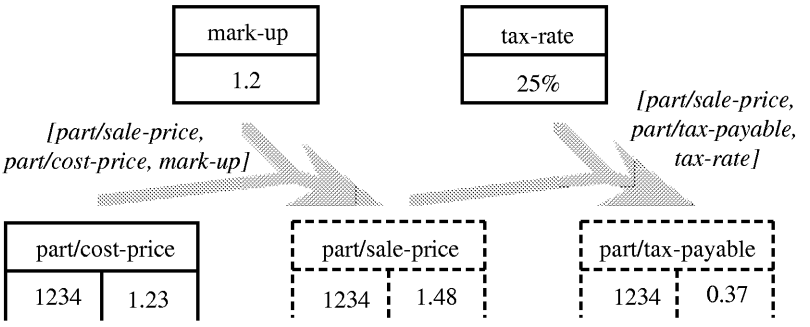


Fig. 1. Real and virtual items

$\lambda^{\alpha}(part)$

part
part-number
1234
2345
2468
3456
3579
4567

$\Lambda^{\tau}(part)$

part
part-number
1234
2345
2456
2468
3456
3579
4680

$\lambda^{\alpha}(part/cost\text{-}price)$

part/cost-price	
part-number	dollar-amount
1234	1.20
2345	2.40
2468	3.60
3456	4.80
3579	5.10
4567	6.30

$\Lambda^{\tau}(part/cost\text{-}price)$

part/cost-price	
part-number	dollar-amount
1234	1.20
2345	2.60
2456	0.60
2468	3.80
3456	4.80
3579	5.20
4680	0.80

Fig. 2. The implementation and the true set

part/tax-payable to be derived. This is illustrated in Fig. 1. If a virtual item A_i is a component of a knowledge item A where the tuples (or data values) associated with A_i are derived from $\{ \lambda^{\alpha_j}(A_j) : A_j$ is a component of $A, j \neq i \}$ using the knowledge A then A_i is *derivable* and those tuples (or data values) are called the *derived set* which is denoted by $\lambda^{\beta}(A_i)$ where β is the time at which the derivation is performed. This definition is recursive. In Fig. 1 only part/cost-price, mark-up and tax-rate are stored. This example shows how the decay of the virtual item *part/tax-payable* is determined by the decay of those three real items *and* by the decay of those two knowledge items. So if any of those three real items or either of those two knowledge items has decayed in some way then that calculation *may* yield an incorrect result.

For a knowledge base, its implementation may have decayed because *updates* that should have been performed were not, or *modifications* that should not have been performed were. The corruption of a knowledge base by such modifications is not considered here. The failure to perform updates is considered. So *updates* are changes that should have been performed on the implementation of real items or knowledge items. In addition, incorrect values may be attributed to a knowledge base implementation because the derived tuples for some virtual items were calculated prior to required updates being performed [8]. At time α the *true set* for a—real or virtual—item A is the set of tuples that should be associated with A at time α; it is denoted by $\Lambda^{\alpha}(A)$. In other words, the implementation is what *is* either stored or derived in the knowledge base, the true set is what *should be* either stored or derived.

Suppose that the implementation of the real data item *part* was stored at time α. Then at a subsequent time τ the implementation and the true set may be as shown on the left side of Fig. 2. In that Figure the implementation for the data item *part* contains the part number "4567" that should *not* be there, and does *not* contain two part numbers which *should* be there. Suppose that the implementation for the real information item *part/cost-price* was stored at time α. Then at a subsequent time τ the implementation and the true set may be as shown on the right side of Fig. 2. Likewise the implementation and the true set of a knowledge item may contain tuples that should not be there, and may not contain tuples that should be there.

4. Knowledge Decay

At time τ, $\Lambda^{\tau}(A)$ and $\lambda^{\alpha}(A)$, $\alpha \leq \tau$, may not be the same. The implementation of a real item is "correct" as long as its tuples have been correctly stored and maintained [9]. The derived set of a virtual item is "correct" if the knowledge used to derive the tuples for that item has been correctly maintained *and* the stored data used by that knowledge has been correctly maintained. In reality we may hope that the implementation is correct, and expect that it is incorrect. To measure the extent that the implementation or the derived set are the same as the true set, let $p_{X/Y}$ be the proportion of those elements in set X that are also in set Y. Then the *difference measure*:

$$\Delta(A, B) = \sqrt{p_{A/B} \times p_{B/A}}$$

is unity if both sets are identical and is zero if one set contains no elements from the other; the square root ensures that this measure retains linearity with measured proportion. For example, if each set contains exactly half of the members of the other set then the value of this measure is 0.5. The value of the difference measure is *not* necessarily equal to the proportion of valid members in either set because the difference measure takes account of those elements that are not in each set but should be there. The *decay* of a real or virtual item A at time τ is:

$$\delta_A(\tau) = \Delta(\lambda^{\tau}(A), \Lambda^{\alpha}(A))$$

where $\tau \geq \alpha$ and the difference measure Δ is as defined above. $0 \leq \delta_A(\tau) \leq 1$. If $\delta_A(\tau) = 1$ then item A is valid. If $\delta_A(\tau) = 0$ then the set of tuples associated with item A contains no tuples that it should contain and A has decayed completely.

If we know precisely what knowledge decay has occurred then we can usually rectify it [6]. In practice we tend to have some loose expectation ε of the decay δ. For example, suppose the knowledge item *[part/sale-price, part/cost-price, mark-up]* is built by applying the knowledge object ***mark-up-rule*** to the three items *part/sale-price*, *part/cost-price* and *mark-up*. That knowledge item may be used to derive tuples for the information item *part/sale-price*. We may expect that "within a year the whole part/cost-price relation will be out of date". So our expectation for the decay of the *part/cost-price* item may be represented by a function with a linear decay of one year's extent. Also, we may expect that "as the 'types' of parts are redesignated, the contents of the part/type relation will decay decreasingly over time so that in a year roughly half of the relation will be out of date and in a 'very long time' the whole relation will be out of date". So our expectation for the decay of the

part/type item is represented by a function with an exponential decay with a half-life of one year. The decay estimates for these two examples are:

$$\varepsilon_{part/cost\text{-}price}(\tau) = \begin{cases} 1 - \tau & if \quad 0 \le \tau \le 1 \\ 0 & otherwise \end{cases}$$

$\varepsilon_{part/type}(\tau) = 2^{-\tau}$ for $\tau \ge 0$

If an object has decayed then such decay will contribute to the decay of any item generated by that object. But objects do not have a value set or an implementation. The decay of the item A, generated using object $\boldsymbol{B}$, $A = \boldsymbol{B}(C, D, E, F)$ may be analysed completely in terms of the decay of items C, D, E and F and the decay of object $\boldsymbol{B}$. In other words, if items C, D, E and F are valid then the decay of item A will be attributable entirely to the decay of object $\boldsymbol{B}$. So the *decay* of an object is the decay of any item generated by that object when applied to a set of valid items.

5. Propagating Decay Estimates

It would be convenient if:

$\varepsilon_{\boldsymbol{ob}}(cc, dd)(\tau) = \varepsilon_{\boldsymbol{ob}}(\tau) \times \varepsilon_{cc}(\tau) \times \varepsilon_{dd}(\tau)$

but this product rule is *not* valid in general because the decay of the components—in the above example the components are *cc* and *dd*—may be logically dependent. This product rule is valid if the decay of the object ***ob*** and the decay of the two component items are all independent. The decay of X is *independent* of the decay of Y if $(\varepsilon_X(\tau) \mid \varepsilon_Y(\tau)) = \varepsilon_X(\tau)$. The decay of X is *determined by* the decay of Y if $(\varepsilon_X(\tau) \mid \varepsilon_Y(\tau)) = \varepsilon_Y(\tau)$.

Suppose that object ***ob*** is applied to a set of n component items $C = \{cc, D\}$ where *cc* is a component item and D is a set of n-1 component items. The general rule for propagating decay estimates through an object operator is:

$$\models \varepsilon_{\boldsymbol{ob}(C)}(\tau) = \varepsilon_{\boldsymbol{ob}}(\tau) \times (\varepsilon_C(\tau) \mid \varepsilon_{\boldsymbol{ob}}(\tau)) = \varepsilon_C(\tau) \times (\varepsilon_{\boldsymbol{ob}}(\tau) \mid \varepsilon_C(\tau))$$

where $(\varepsilon_C(\tau) \mid \varepsilon_{\boldsymbol{ob}}(\tau))$ is "an estimate of the decay of the set of n component items C at time τ given that the estimate of the decay of object ***ob*** at time τ is $\varepsilon_{\boldsymbol{ob}}(\tau)$." If the knowledge base has been normalised then $(\varepsilon_C(\tau) \mid \varepsilon_{\boldsymbol{ob}}(\tau)) = \varepsilon_C(\tau)$. To propagate decay estimates across a set of component items, suppose that the set C is a set of n component items as above. The general recursive rule for propagating decay estimates over such a set is:

$$\models \varepsilon_{\{cc, D\}}(\tau) = \varepsilon_{cc}(\tau) \times (\varepsilon_D(\tau) \mid \varepsilon_{cc}(\tau))$$

where $(\varepsilon_D(\tau) \mid \varepsilon_{cc}(\tau))$ is "an estimate of the decay of the set of n-1 component items D at time τ given that the estimate of the decay of the component item *cc* at time τ is $\varepsilon_{cc}(\tau)$." In general $(\varepsilon_D(\tau) \mid \varepsilon_{cc}(\tau)) \neq \varepsilon_D(\tau)$ even if the knowledge base has been normalised. Now suppose that item *cc* is virtual and that it is derivable from the set of items D using knowledge item ***ob***(cc, D), ie that $\text{Can}(cc, D) = 1$ in ***ob***(cc, D). Then the decay of item *cc* depends on the decay of object ***ob*** and on the decay of the items in the set D. These decay estimates are propagated by:

$$\models \varepsilon_{cc}(\tau) = \varepsilon_{\boldsymbol{ob}}(\tau) \times (\varepsilon_D(\tau) \mid \varepsilon_{\boldsymbol{ob}}(\tau))$$

If the knowledge base has been normalised then $(\varepsilon_D(\tau) \mid \varepsilon_{\boldsymbol{ob}}(\tau)) = \varepsilon_D(\tau)$.

The three rules above for propagating decay estimates follow from the definition of item decay. These three rules lead to complex expressions for decay estimates due to the quantity of conditional expressions—ie expressions involving "|". But the quantity of these conditional expressions may be reduced.

If the knowledge base has been decomposed [10] and if the sub-item relationships have been reduced to sub-type relationships between data items then the object operators and the basis items in the conceptual model are independent with the possible exception of sub-item relationships between data items [5]. So if the knowledge base has been decomposed and there are no sub-item relationships between data items then all the conditional expressions may be removed [1]. For example, if the knowledge base has been decomposed and there are no sub-item relationships between data items then the decay estimate for the virtual information item *part/tax-payable* is:

$$
\begin{aligned}
\varepsilon_{part/tax\text{-}payable}(\tau) &= \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau) \\
&\quad \times (\varepsilon_{\{part/sale\text{-}price,\ tax\text{-}rate\}}(\tau) \mid \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau)) \\
&= \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau) \times \varepsilon_{\{part/sale\text{-}price,\ tax\text{-}rate\}}(\tau) \\
&= \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau) \times \varepsilon_{part/sale\text{-}price}(\tau) \\
&\quad \times (\varepsilon_{tax\text{-}rate}(\tau) \mid \varepsilon_{part/sale\text{-}price}(\tau)) \\
&= \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau) \times \varepsilon_{part/sale\text{-}price}(\tau) \times \varepsilon_{tax\text{-}rate}(\tau) \\
&= \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau) \times \varepsilon_{\boldsymbol{mark\text{-}up\text{-}rule}}(\tau) \\
&\quad \times (\varepsilon_{\{part/cost\text{-}price,\ mark\text{-}up\}}(\tau) \mid \varepsilon_{\boldsymbol{mark\text{-}up\text{-}rule}}(\tau)) \\
&\quad \times \varepsilon_{tax\text{-}rate}(\tau) \\
&= \varepsilon_{\boldsymbol{tax\text{-}rule}}(\tau) \times \varepsilon_{\boldsymbol{mark\text{-}up\text{-}rule}}(\tau) \\
&\quad \times \varepsilon_{part/cost\text{-}price}(\tau) \times \varepsilon_{mark\text{-}up}(\tau) \times \varepsilon_{tax\text{-}rate}(\tau)
\end{aligned}
$$

If sub-item relationships are present or if the knowledge base has *not* been normalised then the calculations become more involved. For example, suppose the *supervisor* data item is a sub-item of the *person* data item. Consider the real *person/supervisor* item; the implementation of which is populated with 2-tuples (person-id, person-id) where the second person is the "supervisor" of the first. Suppose that this item is built by applying the ***super*** information object to the data items *person* and *supervisor*:

person/supervisor = ***super***(*person*, *supervisor*)

then the decay estimate of the *person/supervisor* information item will be:

$$
\begin{aligned}
&\varepsilon_{\boldsymbol{super}(person,\ supervisor)}(\tau) \\
&\quad = \varepsilon_{\{person,\ supervisor\}}(\tau) \times (\varepsilon_{\boldsymbol{super}}(\tau) \mid \varepsilon_{\{person,\ supervisor\}}(\tau)) \\
&\quad = \varepsilon_{person}(\tau) \times (\varepsilon_{supervisor}(\tau) \mid \varepsilon_{person}(\tau)) \times \varepsilon_{\boldsymbol{super}}(\tau) \\
&\quad = \varepsilon_{person}(\tau)^2 \times \varepsilon_{\boldsymbol{super}}(\tau)
\end{aligned}
$$

where $(\varepsilon_{\boldsymbol{super}}(\tau) \mid \varepsilon_{\{person,\ supervisor\}}(\tau)) = \varepsilon_{\boldsymbol{super}}(\tau)$ assuming that the knowledge base has been normalised and the decay of the ***super*** operator is independent of the decay of the items to which it is applied; and where $(\varepsilon_{supervisor}(\tau) \mid \varepsilon_{person}(\tau)) = \varepsilon_{person}(\tau)$ because the decay of *supervisor* is assumed, quite reasonably, to be determined by the decay of *person*.

6. Conclusion

Decay is the degradation of integrity over time [11]. In a unified knowledge representation, data, information and knowledge are all represented in a single formalism. Two measures of knowledge integrity are included in a unified knowledge representation. A join operator is defined that enables items in the unified representation to be joined together to form other items. Item join preserves the validity of the two measures of integrity. Decomposition is defined in terms of join. Knowledge base normalisation is defined in terms of decomposition. Normalisation removes hidden relationships from the conceptual model. Hidden relationships can present a maintenance hazard. The representation of a knowledge base using the unified representation, and the normalisation of that representation simplifies the estimation of knowledge decay.

References

1. Debenham, J.K. "*Knowledge Engineering*", Springer-Verlag, 1998.
2. Debenham, J.K. "Knowledge Simplification", in *proceedings 9th International Symposium on Methodologies for Intelligent Systems ISMIS'96*, Zakopane, Poland, June 1996.
3. Date, C.J., "*An Introduction to Database Systems*" (4th edition) Addison-Wesley, 1986.
4. Debenham, J.K. "Knowledge Object Decomposition", in *proceedings 12th International FLAIRS Conference*, Florida, May 1999, pp203—207.
5. Debenham, J.K. "Understanding Expert Systems Maintenance", in *proceedings Sixth International Conference on Database and Expert Systems Applications DEXA'95*, London, September 1995.
6. Barr, V. "Applying Reliability Engineering to Expert Systems" in *proceedings 12th International FLAIRS Conference*, Florida, May 1999, pp494—498.
7. Mayol, E. and Teniente, E. "Addressing Efficiency Issues During the Process of Integrity Maintenance", in *proceedings Tenth International Conference on Database and Expert Systems Applications DEXA'99*, Florence, September 1999, pp270—281.
8. Katsuno, H. and Mendelzon, A.O., "On the Difference between Updating a Knowledge Base and Revising It", in *proceedings Second International Conference on Principles of Knowledge Representation and Reasoning, KR'91*, Morgan Kaufmann, 1991.
9. Debenham, J.K. "From Conceptual Model to Internal Model", in *proceedings Tenth International Symposium on Methodologies for Intelligent Systems ISMIS'97*, Charlotte, October 1997, pp227—236.
10. Debenham, J.K. "Representing Knowledge Normalisation", *in proceedings Tenth International Conference on Software Engineering and Knowledge Engineering SEKE'98*, San Francisco, US, June 1998 pp132—135.
11. Tayar, N. "A Model for Developing Large Shared Knowledge Bases" in *proceedings Second International Conference on Information and Knowledge Management*, Washington, November 1993, pp717—719.

A Structured Testing Methodology for Knowledge-Based Systems

Abeer El-Korany

Central Laboratory for Agriculture Expert Systems (CLAES), Dokki, Giza, Egypt.
abeer@mail.claes.sci.eg
Ahmed Rafea

Computer science Dept., American university in Cairo, Cairo, Egypt.
Rafea@mail.claes.sci.eg
Hoda Baraka

Computer Engineering Dept., Faculty of Engineering, Cairo University, Dokki, Giza, Egypt.
Saad Eid

Communications and electronics Engineering Dept., Faculty of Engineering, Cairo University, Dokki, Giza, Egypt.

Abstract. In recent years, knowledge-based software technology has proven itself to be a valuable tool for solving hitherto intractable problems. Developers of knowledge-based systems must ensure that the system will give its users accurate advice or correct solutions to their problems. Thus, knowledge-based systems must be debugged and validated just like any other piece of software. It has been found that one of the most important problems in developing knowledge-based systems is the lack of methods to verify and validate its KB. The aim of this article is to define a methodology and its supporting tool set that are used together in order to completely test knowledge-based systems. The suggested testing methodology couples different verification and validation activities that are collectively valuable in raising the level of system correctness.

1. Introduction

One fundamental characteristic of all knowledge-based systems (KBSs) is the clear and clean separation between the knowledge that the system is using and the program that utilize it for problem solving. The two components that compose any KBS are therefore, a knowledge base (KB) and an inference engine (IE). Since the inference engine is algorithmic software i.e., conventional software, software engineering testing techniques can be applied on it. Therefore, in the field of AI testing for KBSs is limited to testing of KB [Vicat, Brezillon & Nottola, 1995]. In fact, the quality of a KBSs is often adequate to the quality of the knowledge stored in the KB [Smith & Kandel, 1993]. Our suggested testing methodology focus on improving the correctness related aspects of the KB that leads to the creation of high quality KBSs.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 427-436, 2000.

The proposed testing methodology is based on the modularity offered by the library-based development approach used in building the KB from reusable component [Abdelhamid, Hassan & Rafea, 1997]. This model that was originally introduced by KADS methodology [Wielinga, Schreiber & Breuker, 1992] makes it possible to derive the structure of a KB. Therefore, it allows each KB component to be tested before its usage (this is analog to unit testing in software engineering). When more that one component are integrated, they should be tested to check the effect of combining them together (this is analog to integration and system testing in software engineering). The methodology is enriched with an integrated tool set to verify and validate the KB that jointly satisfy the production of high quality KBs. Most research in verification and validation (V&V) of KBSs has been performed on rule-based systems [Vermesan, 1998]. This is not to say, however, that systems built with other paradigms do not require V&V. Our proposed tools are able to verify and validate KB contains different knowledge representation like tables and mathematical functions.

Section 2 gives a brief overview of existing V&V techniques for KBSs. Section 3 presents the proposed testing methodology. It has been widely known that KB validation by dynamic testing is an accepted evaluation techniques for achieving KBS quality [Mengshoel & Delab, 1993]. Based on this issue, automatic test case generation techniques, the main contribution of our work, are illustrated in Section 4. Study of the capability of automatic tool to test different KB components is demonstrated by examples in Section 5. Section 6 summarizes the paper.

2. Verification and Validation of KBSs

Ensuring the quality of KBSs involves two types of activity: verification and validation. These two activities are complementary, each is effective at detecting errors that the other will miss, and they are therefore usually employed together [Russby, 1988]. Although V&V are equally important for ensuring the quality of KBS, there is an obvious gap between the current state of art in these two areas. While automatic verification is quite advanced, relatively little work has been done in automated validation [Zlatareva & Preece, 1994]. A set of majors V&V activities has been defined by Vermesan is listed here. Readers who are interest in more details can refer to Vermesan, [1998]. These activities are: competency, consistency, completeness, correctness, testability, usability, relevance, and reliability.

2.1Verification

Verification is “the process of determining whether or not the product of a given phase of software development meets all the requirements established during the previous phase.” [The IEEE definitions (No.72-1983]. Studies have demonstrated that verification can lead to the early detection of errors that otherwise would have remained even after extensive validation tests [Preece, 1995]. Much of the known work in verification of KBSs has been done on automatic tools that check the KBs for the presence of different KBs anomalies. [Ayel & Laurent,1991; O’Keefe, 1993].

2.2 Validation

Validation is concerned with demonstrating whether or not the software product actually solves the customer's problem. Validation is "the process by which delivered code is directly shown to satisfy the original user requirements". [The IEEE definitions, 1983]. The most widely used empirical validation technique is testing [Preece,1995]. This technique involves running a set of test cases on the KBSs and compares the output for agreement with those of an expert or a panel of experts. The main difficulty lies in choosing a set of test problems upon which to measure the completeness of human and machine. In recent years, a number of article and reports have considered criteria and methods for creating a representative suite of test cases [O'Keefe, Balci & Smith, 1987; Chang, Combs & Stachowitz,1990; Zlatareva & Preece,1994].

3. The Proposed Testing Methodology

Traditional methods of software testing involve running test cases through the system and evaluating the correctness of the result obtained. Analysis has shown that these methods are not sufficient for testing KBSs [Ayel, 1991; Laurent,1992]. Since KBSs solve complex problems that have very large input domains', choosing a set of test cases is an extremely difficult problem, and such a set is likely to be very large. Thus, it is worthwhile to look for new testing techniques that are applicable to KBSs. It is assumed that there is not a commonly accepted methodology to verify and validate KBS [Avelino & Douglas, 1993;Vale, 1998]. But several approaches exist that in combination with each other can serve to accomplished the high quality KBSs. Two main features that distinguish our testing methodology: modularity and coupling different V&V activities that are valuable in raising the level of system correctness.

3.1 Modularity

Testing can be applied both to individual system components, as well as to the system as a whole. This is analogue to the idea of unit testing versus integration testing in conventional software engineering which state that "To verify that the system is grossly correct, it is inefficient to begin testing if the basic elements are not correct". The KB can be considered as a compound object, build from subcomponents. Therefore, beginning with test the fundamental components of the KB will prevent errors from propagating throughout the whole KB. We begin by testing all the domain knowledge components, then inference steps, and finally the complete task (bottom-up). Moreover, we can reuse test cases of domain knowledge in testing inference knowledge and test cases of inference knowledge in testing tasks.

3.2 Coupling V&V Activities

The suggested testing methodology can be viewed as possessing two testing activities: static and dynamic. Static testing (verification), which does not involve the execution of KB, examines internal correctness and consistency of the KB. Dynamic testing (validation) is the execution of the KB on a set of well-defined test cases to evaluate the functional, structural or computational aspects of the system. The suggested testing methodology couples V&V activities that comply with the ideal component for V&V presented in the section2. Consistency and completeness are achieved by verification. While validation covers competency, correctness, and reliability of the system. Validation of the 'usability' of the system is out of our scope. Relevance is not considered here.

3.2.1Verification. The verification process of the KB ensures that the system is free of induced errors by the developer as well as ensuring matching between outputs of different development phases. We provide two main verification activities.

3.2.1.1 Automatic Verification. An automatic verification tool has been developed to detect consistency, completeness and other error of the KB. Although verification is the keystone of the testing process, analysis of the automatic verification process is not considered in this article, readers who are interest in more details may refer to [El-Korany, Shaalan, Baraka, Rafea, 1998].

3.2.1.2 Human Analysis. These techniques usually rely on individuals to use their expertise to find errors in the KB. Review, walkthrough and inspection are the techniques that are used to verify the knowledge. They are limited to number of errors that could be detected "by eye" like consistency between different development life cycle (consistency between requirement specification, design and implementation).

3.2.2 Validation. Assuming that the KB has been verified, validation ensures that the knowledge it contains correctly represents and simulates the domain knowledge. Two aspects constitute the validation task in our methodology: test case generation methods and regression testing.

3.2.2.1 Test Case Generation Methods. We have to define two types of tests for each KB component, automatic test and developer test.

Automatic test: Test cases are automatically generated in the form of input concept-attributes pair and their suggested values. These suggested inputs are applied to the component to be tested and the output is ranked. The tester provides both the knowledge engineer and the domain expert by a list of test cases. This list of test cases serves two functions. First the knowledge engineer compares it with the requirement specification to check consistency between them. Second, the domain expert ensures the quality of the solution (whether they meet the required knowledge or not). *Developer test*: By this test the developer will be able to randomly test his system. He can run different KB components independently. A screen holding the input concept-attributes pairs used in component to be tested is automatically generated and displayed. This screen contains the possible legal values (in the case of nominal attribute) or its boundary (in the case of numerical attribute) for each attribute. The existence of domain expert will enrich this test since he could apply different combination according to his expertise. This test ensure the robustness of the system (the ability of the system to solve any conflict that my appear due to different combination of input [Mazas, 1991]).

3.2.2.3 Regression Testing. Regression testing is an important part of the overall validation process since it aims to determine whether a product has regressed to a less functional state than in the previous build [Shafer, 1998]. The dynamic testing strategy of regression is used after modification. Since the probability of making an error while introducing a change is between 50 and 80%. In the use of traditional regression, all previous test cases are reapplied. This leads to an expensive overhead to impose on a testing scheme. To overcome this drawback, we develop a composite test case library contains old test cases that have been applied to each KB components. When any component is modified, it is the only one that is re-tested. Comparison between the old and the new cases is made. The expert decides whether there is a fault or the differences are expected.

4. Automatic Test Case Generation Techniques

The most practical method for creating a set of test cases for KB would seem to entail a combination of both structural and functional approaches [Zlatareva & Preece,1994; Rushby, 1988]. Based on this criterion, each KB component should have its own testing techniques according to its role and representation. Functional test is applied to all domain knowledge while a structural testing guided by functional test is proposed to both inference steps and subtasks. Test cases have the following structure

1. A combination of defined inputs for the tested component.
2. Expected output achieved by the execution of such KB component under inputs specified in 1.

4.1 Domain Knowledge Testing

4.1.1 Rule Testing. Each rule consists of condition part and an action part. A condition part consists of a set of premises. A premise is in the form of: Concept, attribute, operator, and value. Verification of a rule is able to discover the following errors in a premise: nonexistence concept/ attribute /value, value are out of range or inconsistency with definition, and inconsistency combination of concept/attribute/value. While other errors in a premise like: incorrect concept/ attribute name (not meet specification), incorrect value (less or more) and incorrect operator can only be discovered by testing. Discovering other errors in a rule like: premises combined with wrong Boolean logic, missing premise, incorrect action part, and overall effect of rule is illogical W.R. requirement can only discovered by testing.

The principle underlying our technique is that each rule should be fired at least once in order to obtain a complete coverage with getting over all expected errors. Therefore, we generate two cases for each rule one exactly true and the other is minimally false. The true case is the combination of true values for each rule condition, while the false case is the combination of false values for each rule condition. True value is the one just inside the true interval, computed as the attribute value in case of '=', '>=', or'<=' . While it computed as the attribute value plus or minus the smallest step interval in case of '!=', '>', or '<' operators. False value represents the attribute value in the immediate vicinity of the false boundary

computed as the value in case of '!=', '>', or'<' . While it computed as the value plus or minus the smallest step interval in case of '=', '>=', or '<=' operators.

4.1.2 Table Testing A table is represented in the form that is very close to tables' concept used in the context of database. The difference is that while database fields contain absolute values, tables entries may contain logical expressions. Thus, a table row is similar to a rule but with fixed inputs and outputs. In software engineering, the equivalence class partitioning divide the input space into classes of input that are expected to produce similar output. The effectiveness of equivalence class analysis could be applied on table testing. Since each of the table rows contains the same input/output concept/attribute pair but with different values. The table test case generator produce test cases that would probe table inputs at least once.

4.1.3 Function Testing Function representation is usually used when the relation can be represented in the form of a mathematical equation. History shows that most software failures are detected by a tester or an end user applying an input at its first or last, or highest or lowest valid values (i.e., boundaries). Since function used numeric values (if any other types are involved, it must have been detected by verification). We select three test cases for a function: one default, and two for boundary condition.

4.2 Inference Knowledge Testing

An inference step is a collection of domain relations that are needed for achieving a

Fig. 1. Testing algorithm for an inference step

```
IS:= inference step of the KB;
Rs:= List that contains rule
cluster of IS;
TFs:= List that contains
function and tables of IS;
Begin
 For each element of Rs do
 Begin
  Generate test cases for
  Rs[I] such that each rule
  in Rs[I] has one test case;
 End
 Sort Rs in descendant order
 according to the number of
 generated test cases;
 Get Rs[1];
 Len:= Length of Rs[1];
 Accumulate test cases of
 other rule clusters on Rs[1];
 For each element of TFs do
 Begin
    Generate test cases for
    TFs[i] equal to Len;
 End
 Accumulate test cases of
 tables and functions on Rs[1]
End
```

Main testing algorithm

```
L:= List that contains rule clusters of inference step IS
sorted in descendant order ;
L1:= List that contain function and tables of IS;
Len := Length of L;
Num := Number of test cases of L[1];
Begin
 For all element in L1 do
    Generate test cases equal to Num;
 TC:= List that contain test cases of L[1];
 For I:=2 to I:= Len do
 Begin
  TC1= List that contain test cases of L[I];
  N :=Number of test cases of L[I];
  For j:=1 to j:= n do
  Begin
   C:= diff between conc-attribute values of TC1[j],TC[j];
   Append(TC[j],C);
   j:=j+1;
  end
  I:=i+1;
 End
 For i:=1 to i:= Len do
 Begin
  TC2= List that contain test cases of L1[i];
  For j:=1 to j:= Num do
  Begin
   C:= diff between conc-attribute values of TC2[j],TC[j];
   Append(TC[j],C);
   j:=j+1;
  end
  I:=i+1;
 End
End
```

Algorithm to accumulate test cases of inference step

domain specific function at the knowledge level. Moreover, there is no internal dependency between the domain relation that the inference step contains. At the

inference layer integration test takes place. The main concern of such test is to find out the effect of combining these components. Considering function testing, if we have an inference step with **N** inputs we could get **one** test case with **N** parameters. In this case it could happen that neither of its component be fired (rule, table or function) and thus this test case is useless. Considering structure testing, each component should have its separate test cases since they are independent. Thus, for an inference step contains $\{R_1,.....R_M\}$, R_1 contains n_1 rules, R_2 contains n_2 rule, etc., and we can get $n_1+n_2+n_3....+n_M$ test cases plus one case for each table and one case for each function. This will cover the structure of the inference step but with large number of test cases. Another drawback of structure testing is that we will not be able to measure the integration between different knowledge component. We combine the advantage of both testing strategies in one adapted technique with the goal of overall effectiveness in mind. For an inference step contains **M** relation and the maximum relation length is **N** we reduce the number of the test cases to be equal the number of rule for the maximum relation length keeping 100 % coverage. Figure 1 illustrates the main algorithm used to generate test cases of an inference step as well as the accumulation algorithm that is considered the core of this test case generation method.

4.3 Task Knowledge Testing

A task is the part of the KB, that represents its procedural knowledge. Actually, a task structure can be thought of as a control structure over a collection of transfer tasks, sub-tasks, and procedures. Each subtask consists of a set of inter-related component. Each of these components may refer to inference steps or transfer tasks, or procedures or others sub-tasks. Test case generation process of a subtask is somehow similar to that of inference step (since both of them represent a collection of other KB components). However, the components of the subtask are dependent on each other in opposite to that of inference step. The dependency between task components means that outputs of one component permits the other component to be fire-able. In other words, inputs of each subtask are either driven from transfer tasks or provided by any previous subtasks. Thus, the process of generating test cases of subtasks must cover testing of its components. Inputs of a subtask are not a combination of its components but the algorithm of task testing determines these inputs. Values assigned to these inputs are estimated according to the following steps.

1. Generate test cases of each inference steps of this subtask.
2. Sort inference steps in descendant order. (According to number of generated cases).
3. Get inference step with max number of cases.
4. Accumulate test cases of other inference steps on this inference
5. Determine subtask input.
6. Extract subtask inputs from the accumulated test case in step5

Algorithms used to generate test cases of a sub- task are illustrated in fgure2.

Fig. 2 Testing algorithms for a sub-task

```
L:= List that contain IS of task T;
Len := Number of IS of task T;
Begin
  For I:=1 to i:= Len-1 do
  Begin
   N:= num of generted test cases of L[i];
   For j:=i+1 to j:= Len do
   Begin
    n1:= number of generated test cases of
    L[j];
    If n <n1 then
    Swap( L(i),L(j));
    j:=j+1;
   end
   I:=i+1;
  End
End
```

Sort IS in ascending order

```
L:= List that contain IS of task T;
Len := Number of IS of task T;
Begin
   For I:=1 to I:= Len do
   Begin
   j:=1;
   IN:= input attribute of L[i];
   Append (OUT,output attribute of L[i])
   For all element in IN do
   Begin
     If IN[j] is drived from transfer task
   then
     Append( L,IN[j]);
     Else If IN[j] not appear in OUT then
     Append(L,IN[j]);
     j:=j+1;
   end
   i:=i+1;
end
```

Algorithm to determine task input

```
L:= List that contain IS of task T;
Len := Number of IS of task T;
Begin
  TC:= List that contain test cases of L[1];
  For I:=2 to I:= Len do
  Begin
    TC1= List contain test cases of L[I];
    Num :=Number of test cases of L[i];
    For j:=1 to j:= num do
    Begin
      C:=diff between con-att values of
      TC1[j],TC[j];
      Append(TC[j],C);
      J:=j+1;
  End
  I:=i+1;
  End
End
```

Algorithm to accumulate test cases of IS of task

```
TC:= List that contain accumulated test cases
of task T;
IN:= List that contain input concept attribute
of task T;
Num :=Number of test cases of TC;
Test :=empty list that will hold test case;
Begin
    For I:=1 to i:= Num do
    Begin
    Diff:= extract all IN members that appear
    in TC[i];
    Append(Test[i],Diff);
    I:=i+1;
    End
End
```

Algorithm to select task inputs from test cases

5. Examples of Utilization and Testing

Several examples covering automatic test case generation of different KB component are illustrated here. These examples demonstrate the capability of our tool to generate different test cases for each KB component according to its role and representation. The examples presented here were taken from irrigation KB of cucumber crop management under plastic tunnel [Rafea &El-Azhari & Ibrahim & Edres & Mahmoud 1995], which is developed by Central Laboratory of Agricultural Expert Systems (CLAES).

5.1 Result of Domain Layer Testing

The automatic test case generation tool works on different domain layer components in turns. A list that contains names of these components appears to the tester, who freely selects the one that he would like to test. An important point to note here is that, although each KB component is verified, some error still exist that are

discovered by test cases. This is best explained by an example. Consider the following function that is used to calculate the actual available water for the soil by subtracting two soil parameters. soil.aw_p = soil.fc-soil.pwp

If the first operand is less than the second operand, we obtain negative result that contradicts with the requirements. To avoid this, a condition that prevents negative output should be added as a premise of this function. The automatic test cases allow us to identify this error that could never be detected by any other tests as shown in figure3.

Fig. 3. Automatic test case generated for function soil.aw_p

Test cases of function soil.aw_p		
'input is.'. ' soil.pwp= 978.0.'. ' soil.fc= 265.0.'. 'ouput is.'. @soil.aw_p= -713.0.'.	'input is.'. ' soil.pwp= 0.001.'. ' soil.fc= 0.001.'. 'ouput is.'. ' soil.aw_p= 0.0.'.	Input is.'. ' soil.pwp= 1000.0.'. ' soil.fc= 1000.0.'. 'ouput is.'. ' soil.aw_p= 0.0.'.

5.2 Result of Inference and Task Layer Testing

The goal of testing both inference and task layer is to measure the effect of integrating KB components in order to achieve a specific function. The suggested testing techniques for inference and task layer aim at reduce the number of required test cases with the goal of complete coverage in mind. For example in the irrigation system, we have a sub-task "Calculate irrigation interval" that has 4 inference steps with different inputs, the proposed task testing technique generate only one case to test the integration of this subtask components.

6. Conclusion

We developed a testing methodology with its supporting integrated tool set that are used to test KBSs. This testing methodology couples different V&V activities. Applying verification activities before validation strengths the testing process. Since semantic errors will be eliminated as early as possible to avoid expensive correction later. For automatic validation we provide different testing techniques that applied for different knowledge types in order to examine different aspect of the test component. These techniques are collectively valuable in raising the level of system correctness Functional test is applied to all domain knowledge (rule clusters, tables, and functions) while structure guide by functional testing was applied to both inference steps and subtasks. The issue that has been considered is the coverage of test cases. After automatic test takes place, these test cases are applied to human experts to decide whether it meets the specifications. These cases also serve in matching between the requirement specification and the developed system. When a KB component modified regression testing takes place. A comparison between the old and the new cases is applied to the developer to record the effect of modification.

References

1]Abdelhamid,Y., Hassan, H., Rafea, A.,[1997] "An Approach to Automatic KBS Construction From Reusable Domain-Specific Components" Proceedings of the 9th International Conference on Software Engineering & Knowledge Engineering (SEKE'97), Madrid.

2] Avelino J. G., Douglas D., [1993].“The engineering of knowledge-based systems,Theory and practice”,Prentice-Hall International Edition.A simon & schuster company, New Jersey.

3] Ayel,M., Laurent, J.P. [1991]”Foreword Validation, Verification, and test of knowledge-based systems”, John Wiley & Sons, New York.

4]Chang, C. L, Combs, J. B. and Stachowitz R. A.[1990]. “A report on the Expert Systems Validation Associate (EVA)”. Expert Systems with Applications (US), 1(3):217—230.

5]El-Korany A., Shaalan K., Baraka H., Rafea A.,[1998]. " An Approach for Automating the verification of KADS-based Expert Systems”, New Review of Applied Expert Systems, Vol. 4, pp. 107-124, Taylor Graham, UK.

6]Laurent, J.P.,[1992] ”Proposal for Valid Terminology in KBS validation”, Proc.10th European Conf. on Artificial intelligence. John Wiley & Sons, New York.

7]Mazas,P.,[1991]. “Design knowledge validation through experimentation: the SYSIFE system”. In M.yel & P.laurent (Eds.) Validation, verification and test of knowledge based systems (pp.119-145). 1

8]Mengshoel, O. J., Delab,S. [1993] ”knowledge validation: principles and practice” IEEE expert.Leibiitz (Eds.), Managing expert systems. Harrisburg, PA: Idea Group.

9] O'Keefe R. M., O'Leary D. E.,[1993]. “Expert systems verification and validation: A survey and tutorial”. Artificial intelligence Review 7, p3-42.

10]O’Keefe,R.M.,Balci,O.,Smith,e.p,[1987]. “Validating expert system performance”. IEEE expert,Vol.2,No.4,winter,pp 81-90

11]Preece, A.D, [1998]. Building the Right System Right. AAAI-98 Workshop on Verification and Validation of Knowledge-Based Systems, Technical Report WS-98-11, AAAI.

12] Preece,A.D, [1995]. “Towards a Quality Assessment Framework for Knowledge-Based Systems”. Journal of Systems and Software, 29(3), 219-234.

13] Rafea, A., El-Azhari,S., Ibrahim,I., Edres,S., Mahmoud,M., [1995] "Experience with the development and Deployment of Expert Systems in Agriculture" Proceedings of IAAI-95 Conference, Montreal, 19 - 25 August 1995, Canada.

14] Rushby,J.,[1988] ”Quality measures and assurance for AI software ”,(NASA Contractor report CR-4187).Menlo ParkCA:SRI International.

15] Shafer. J ,[1998]. “Regression Testing Basics”.

16]Smith,S.& Kandel, A.,[1993].“Verification and validation of rule-based expert systems” CRC Press.

17] Vale ,A., Ramos,C.m Fernanda,M.,[1998]. “Knowledge based systems for power system control centers”. Validation and verification. Proceeding of 1998 European meeting on Validation and Verification of KBSs ,Trento, Italy.

18] Vermesan A. [1998], "Foundation and application of expert system verification and validation" The handbook of Applied expert system

19] Vicat,C. , Brezillon,P. & Nottola,C.,[1995]. “Knowledge validation in the building of knowledge based systems”. Expert system with application, Vol.8,391-397.

20] Wielinga B. J. and Schreiber A. Th. ,and Breuker, Eds, [1992]. “KADS: A modeling approach to knowledge engineering, Knowledge Acquisition”. (Special issue: The KADS approach to knowledge engineering). March, 4(1) pp. 5-53.

21] Zlatareva,N., and Preece,A.,[1994]. “State of the art in automated validation of knowledge-based systems”. Expert Systems with Applications, 7(2), 151-167.

Semantic Verification of Rule-Based Systems with Arithmetic Constraints

Jaime Ramírez[1], Angélica de Antonio[2]

[1]Universidad Antonio de Nebrija
Escuela técnica y superior de Ingenieros en Informática.
Campus de la Dehesa de la Villa. c/Pirineos, 55. 28040 Madrid, Spain.
e-mail: jramirez@unnet.es

[2]Universidad Politécnica de Madrid
Facultad de Informática
28660 Boadilla del Monte, Madrid, Spain
e-mail: angelica@fi.upm.es

Abstract. The aim of this paper is to show a method that is able to detect a particular class of semantic inconsistencies in a rule-based system (RBS). A semantic inconsistency is defined by an integrity constraint. A RBS verified by this method contains a set of production rules, and each production rule comprises a list of arithmetic constraints in its antecedent and a list of actions in its consequent. An arithmetic constraint is a linear inequality defined in the real domain that includes attributes, and an action is an assignment that changes an attribute value. As rules are allowed to include actions of this kind, the behaviour of the verified RBS is non-monotonic. The method is able to give a specification of all the initial fact bases (FB), and the rules from these initial FB that would have to be executed (in the right order) to cause an integrity constraint to be violated. So, the method builds an ATMS-like theory. Moreover, the treatment of arithmetic constraints is inspired by constraint logic programming.

1.Introduction

The purpose of this paper is to illustrate a method of verifying rule-based systems (RBS). This method actually focuses on detecting semantic inconsistencies. A semantic inconsistency is defined by an integrity constraint (IC). One of the most interesting facets of this method is that it is able to deal with production rules and ICs that contain *arithmetic constraints on attribute values*. To the best of our knowledge, there is no other method or tool that also addresses arithmetic constraints. In addition, as the consequent of the production rules is allowed to change the current attribute values, the behaviour of the verified RBS is *non-monotonic*.

The method builds an ATMS-like theory [4] defined by a set of labels (also called contexts) like COVADIS [10], KB-REDUCER [5] or COVER [7]. Each label describes a set of deductive paths, by means of which to deduce a literal, and a specification of the environment, by means of which to fire each deductive path.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 437-446, 2000.

Moreover, the treatment of arithmetic constraints is inspired by constraint logic programming (CLP) [3,6]. As in some constraint logic programming languages, for example, the Simplex Method [2] is used to check whether a set of real domain constraints could be satisfied.

This method is part of a more ambitious method, called MECORI [8,9]. This method is able to detect semantic inconsistencies in RBSs whose fact bases (FB) contain a lot of object types: frame classes and instances, relationships, propositions, and attribute values.

The section 2 of this paper explains some points related to the kind of RBSs and ICs that are verified by the method. In section 3, the context concept and the operations on contexts that will be used in the subsequent sections are defined. Section 4 explains the procedure for detecting an inconsistency and describes how this inconsistency can be reached in a given RBS. In section 5, we highlight some problems with regard to the treatment of constraints in several domains. We end with some conclusions about our work.

2. Scope of the Method

2.1. Static Aspects

Our method is able to verify a RBS that is formed by a set of propositional logic-based production rules and a declaration set. The declaration set contains the attribute type declarations for each attribute that is used in any rule. Due to the efficiency considerations discussed in section 4, we consider only the real domain for the attributes at this stage.

The production rule form is: R: $c_{11}, c_{12},\ldots, c_{1w} \vee c_{21}, c_{22},\ldots., c_{2v} \vee ,\ldots, \vee c_{m1}, c_{m2},\ldots., c_{ms} \rightarrow a_1, a_2,\ldots., a_t$, where the antecedent part contains a disjunction of *m* conjunctions of arithmetic constraints (c_{ij}), and the consequent part contains a list of assignments (a_k).

Within the scope of this paper, an *arithmetic constraint* is a linear arithmetic inequality defined in the real domain. However, the predicate of a constraint can be either: *is-equal-to* (=), *is-less-or-equal-to* (≤) or *is-greater-or-equal-to* (≥). The reason of this is that the Simplex Method will work only with this kind of constraints. For example, $5a - 3b \geq 7c$ is a valid arithmetic constraint, where *a*, *b* and *c* are attributes.

An assignment is an action that changes an attribute value. The syntax of an assignment is: *attribute_name := real_expression*. The real expression must be linear, and it could include some attribute that also appeared in the antecedent part. For example, $a := b+5*d$ is a valid assignment, where *a* and *d* are attributes. Some rules could incorporate some *recurrent assignments* $a := f(a)$, where *f* is a function that depends on the attribute *a*. We say that an attribute *a* is *deductible* in a RBS iff there

is any rule in the RBS that contains an assignment that changes the value of *a*. An attribute that is not deductible in the RBS is called an *external* attribute. At the beginning of the RBS execution, the values of the external attributes are entered into the RBS as inputs. The RBS then outputs the value of some deductible attributes.

An IC defines a semantic inconsistency by means of a set of arithmetic constraints. The IC form is $c_1, c_2, \ldots, c_n \rightarrow \perp$

2.2. Dynamic Aspects

The RBS is assumed to execute by forward chaining, according to given conflict set resolution criteria. Explicit control mechanisms and meta-rules are not considered by the method. When a rule is fired, we assume the parallel execution of all the assignments belonging to the consequent of the rule. If a rule comprises a recurrent assignment, the result of its firing will differ depending on the number of times that the rule is fired. We will discuss how the method deals with this kind of rules later.

3. Contexts and Context Operations

In this section, we will first define the concept of *context* managed by the method. Next, we will define some operations on contexts. Finally, we define the concept of *context associated with a constraint*.

3.1. Context

The objective of our method is to output a specification of all the initial FBs, and the rules of these initial FBs that would have to be executed (in the right order) in order to produce a violation of an IC. An environment describes a set of initial FBs, whereas a deductive path describes a list of rules.

* An ***environment*** $E_i = \{c_i\}$ is the specification of the set of all the valid initial FBs, in which, at least, all the arithmetic constraints included in E_i are certain.
* A ***deductive path*** $DP_i = \{r_i\}$ associated with an environment E_i is defined as a list of rules, whose firing in the specified order is possible, provided that the arithmetic constraints included in E_i are certain, and leads to the attainment of a given goal.
* A ***context*** C consists of a set of pairs (E_i, DP_i): $C = \{ (E_i, DP_i); i=1,...,n \}$, where E_i is an environment and DP_i is a deductive path associated with E_i.

A *goal* is defined as a set of arithmetic constraints. If *C(h)* is the context associated with a goal *h*, then any pair *(E, DP)* contained in *C* satisfies the following proposition: after firing the rules included in *DP,* from *E*, the goal *h* will be true. We denote this as: $E \Rightarrow^{DP} h$. Formally: *If* $(E, DP) \in C(h)$ *then* $E \Rightarrow^{DP} h$

If the context associated with an IC is empty, this means that such a constraint is always satisfied for all valid initial FBs. On the other hand, if the context associated with an IC is not empty, then each pair included in the context represents a possible IC violation during RBS execution.

In addition, we say that an environment E_i is subsumed by another environment E_j, denoted as $E_j \subset E_i$, iff:

$$\forall a \text{ (attribute)} \in c_k \text{ s.t. } c_k \in E_j \text{ then } (\exists c_l \in E_i \text{ s.t. } a \in c_l \text{ and projection}(a, E_i) \subset \text{projection}(a, E_j))$$

where the *projection(a, E)* set is defined as the set of values for the attribute *a* in the different solutions of the set of arithmetic constraints *E*. The computational complexity of the algorithm that outputs the *projection(a, E)* set is $\Omega(2^m)$, assuming that $n \geq 2m$, where *m* is the number of constraints in *E* and *n* is the number of attribute values in *projection(a, E)* [2].

We consider an arithmetic constraint to be (geometrically) *redundant in an environment* iff it can be removed from the environment without changing the solution set associated with the environment. This latter is expressed formally as: *c is redundant in E iff solutions(E)=solutions(E-{c}),* where the *solutions(E)* set is defined to be the set of all the solutions (attribute value sets) that satisfy all the constraints in *E*.

As mentioned in the previous section, the result of firing a rule with a recurrent assignment depends on the number of times that it is fired. Hence, it is also necessary to somehow represent the number of times that a rule is fired in a deductive path. In order to do so, we associate a variable with the name of the rule in the deductive path. This variable refers to the number of times that the rule is fired in the deductive path. We will call this kind of variables *rule variables*.

3.2. Operations on Contexts

The following operations are used during the context computation process:

3.2.1. Combination of a List of Contexts

Let $C_1, C_2, \ldots, C_n$ be the list of contexts, and $\text{Comb}(C_1, C_2, \ldots, C_n)$ be the context resulting from the combination. The form of this resulting context is:

$$\text{Comb}(C_1, C_2, \ldots, C_n) = \{ (E_{k1} \cup E_{k2} \ldots \cup E_{kn}, DP_{k1} * DP_{k2} \ldots * DP_{kn}) \text{ s.t. } (E_i, DP_i) \in C_i \text{ and order}(k1,k2,..,kn) \}$$

where *order(k1,k2,..,kn)* is a predicate that is only true for a unique permutation of (1,2,..,n). The aim of this predicate is to ensure that the method will yield the same deductive path by combining the same deductive paths in different orders. We need to

define two additional operations in order to implement the context combination operation: the union of environments and the combination of deductive paths:

b.1) ***Union of environments*** ($E_i \cup E_j$): this operation consists of the union of the sets of constraints E_i and E_j. After the union of two sets of constraints, it is necessary to check whether the resulting set of constraints can be satisfied or, in others words, whether the resulting set of constraints is feasible. The Simplex Method and, in particular, the first phase of the two-phase Simplex Method [2], is applied to run this test.

If the resulting environment is a subset of any environment included in the context associated with an input IC, then this pair will be discarded from the combination context, since it is a specification of an invalid starting situation. Otherwise, the resulting environment is considered consistent. Moreover, after combining a pair of environments, there might be some redundant arithmetic constraints in the resulting environment. These redundant arithmetic constraints should be detected and removed from the resulting environment.

b.2) ***Combination of deductive paths*** ($DP_i * DP_j$): let DP_i and DP_j be deductive paths, then $DP_i * DP_j$ is the deductive path that results from appending the lists DP_i and DP_j. Before combining two deductive paths, it is necessary to check whether there is a pair of contradictory actions A and B, such that $A \in DP_i$ (A appears in the consequent of any rule of DP_i) and $B \in DP_j$. *Two actions are contradictory* if they assign a different value to the same attribute. If there is a pair of contradictory actions in DP_i and DP_j, then the pairs (E_i, DP_i) and (E_j, DP_j) will not be combined.

3.2.2. Concatenation of a Pair of Contexts

Let C_1 and C_2 be the pair of contexts, and $Conc(C_1,C_2)$ be the context resulting from the concatenation, then:

$$Conc(C_1,C_2) = C_1 \cup C_2$$

After combining or concatenating a pair of contexts, some redundant pairs could appear in the resulting context. A pair $(E, DP) \in C$ is *redundant* if there exists another pair $(E', DP') \in C$ such that $E \subset E'$. Thus, if a pair $(E, DP) \in C$ is redundant because there exists another pair $(E', DP') \in C$ such that $E \subset E'$, then the pair (E, DP) must be replaced by (E', DP). If $DP=DP'$, then we must remove the pair (E, DP).

3.2.3. Substitution

A substitution is a function, defined by a deductive path, which can be applied to constraints. The idea is to apply all the actions in the rules included in the deductive path to the attributes included in the constraint.

Let us look at an example of substitution:

Let DP=[R2;R1] where
R1: a<1, b+c>7 → d:=5, e:=b+a
R2: d<10, e+a >7 → x:=3d-e
then,
Subst(x<1, DP) = Subst(Subst(x<1, [R2]), [R1]) =
= Subst(3d-e<1, [R1]) = (3*5–(b+a)<1) = (14<b+a)

If an action is recurrent, the effect of the action on an attribute is represented by a *Finite Difference Equation*, taking the form:

$$a(n) = \begin{cases} a_0 & n = 0 \\ b * a(n-1) + g & n > 0 \end{cases} \quad (1)$$

where $a(n)$ is the value of the attribute a after firing the rule n times; a_0 is the value of the attribute before firing the rule; b is a constant; and g is an arithmetic expression that contains other attributes and/or some constants. The general scheme for solving an equation like the one above is as follows:

$$a(n) = b^n a_0 + g\frac{b - b^n}{1 - b} + g \qquad b > 1 \quad (2)$$

$$a(n) = a_0 + gn \qquad b = 1$$

After applying a substitution to a constraint, the resulting constraint will only contain some external attributes and, sometimes (if there are recurrent actions), some rule variables.

If the method needs to find out whether an environment with some rule variables $\underline{x} = (x_1, x_2,\ldots,x_n)$, is feasible in any step, then the method must first test the environment, where $\underline{x} = (1,1,\ldots,1)$. Then, if the environment is not feasible, the method must test the case $\underline{x} = (1,1,\ldots,2)$ and so on, until a feasible environment is yielded. Before executing the method, the maximum number of iterations should be set by the knowledge engineer who designed the RBS. If the process reaches the maximum number of iterations, then the current environment would be assumed not to be feasible.

3.3. Context Associated with a Constraint

The method must generate the context associated with every constraint belonging to the goal to get the context associated with a goal (set of constraints). Moreover, the *context associated with every deductible attribute* belonging to the constraint has to be output to generate the context associated with a constraint.

The context associated with a deductible attribute *a*, denoted as *C(a)*, contains all the pairs *(E, DP)* such that the first rule in *DP* (in order of addition; thus, the leftmost rule in the list of rules) comprises an assignment that changes the value of the attribute *a*. Hence, *C(a)* describes all the initial situations which lead to any modification of the attribute *a*.

Thus, the context associated with a constraint *c* that contains at least one deductible attribute is defined formally as:

$C(c) = \{ (E_1, DP_1), (E_2, DP_2),.., (E_n, DP_n) \}$
where $\forall (E_i', DP_i') \in Comb(C(a_1), C(a_2),\ldots, C(a_m))$ *s.t.* $a_1, a_2,.., a_m$ *are all the deductible attributes included in c, then*
$DP_i = DP_i'$
$E_i = E_i' \cup Subst(c, DP_i')$

If the constraint only contains external attributes, then the context associated with the constraint is: $C(c) = \{ (\{c\}, \varnothing) \}$.

Each pair (E_i', DP_i') belonging to $Comb(C(a_1), C(a_2),\ldots, C(a_m))$ describes a possible way to deduce some value for each deductible attribute belonging to the constraint *c*. Each environment E_i' specifies several relationships among the external attributes and other variables, which must hold true in the initial FB. In addition, the constraint $Subst(c, DP_i')$ specifies the same relationships as *c*, albeit in terms of external attributes and other variables. Therefore, if the constraint $Subst(c, DP_i')$ is added to E_i', then the resulting environment E_i will contain all the constraints, in terms of external attributes and other variables, such that $E_i \Rightarrow^{DPi} \{c\}$. In some cases, the resulting environment E_i could turn out to be unfeasible and should be discarded.

4. Operation of the Method

In this section, we show the steps that are necessary to build the context associated with an IC by means of a process specification and an example of its application.

Process for outputting the context associated with an IC
1.FOR each constraint in the IC:
    1.1.**Output the context associated with a constraint.**
2.Combine the contexts associated with all the constraints ⇒ context
3.RETURN context

Process for outputting the context associated with a constraint
1.FOR each attribute in the constraint:
    1.1. Filter the rules that change the attribute value ⇒ conflict set.
    1.2. Remove from the conflict set the rules that have already been used in the current DP
    1.3. IF conflict set is empty
    THEN (* the attribute is an external attribute*)
        1.3.1. {({constraint}, ∅)} is the context associated with the attribute
    ELSE (* the attribute is a deductible attribute*)
        1.3.1. FOR each rule in the conflict set:

1.3.3.1.**Output the context associated with the rule** ⇒ rule_context
1.3.3.2.Add the rule to every deductive path in rule_context
1.3.2. Concatenate the contexts associated with all the rules included in the conflict set⇒ attribute_context
1.3.3. Attribute_context is the context associated with the attribute
2.Combine the contexts associated with all the attributes ⇒ context
3.FOR each pair (E, DP) in the context:
3.1. Apply the substitution defined by DP to the constraint ⇒ new_constraint
3.2. Add the new_constraint to E.
⇒ new_context
4.RETURN new_context

Process for outputting the context associated with a rule
1. FOR each conjunction in the rule antecedent:
1.1.FOR each constraint of a conjunction:
1.1.1. **Output the context associated with a constraint.**
1.2. Combine the contexts associated with all the constraints included in the conjunction.
2. Concatenate the contexts associated with all the conjunctions included in the rule antecedent ⇒ context
3.RETURN context

These three processes represent a backward chaining simulation of the real rule firing. The recursive calls finish when, in the process of outputting the context of a constraint, an attribute is external. The goal of the step 1.2 in the second process is to avoid cycles in the deductive paths.

After outputting the context associated with an IC, *C(IC)*, any pair (*E, CD*) belonging to *C(IC)* such that *CD* could not fire in a real execution of the RBS according to the conflict set resolution criteria will be removed from the *C(IC)*. So, the method takes into account the conflict set resolution criteria.

Next, we show an example of the application of the method, according to the processes discussed above:

Let A be a RBS defined as A = ({R1, R2, R3}, {a, b, d, e}) where:

R1: $e \geq 1, d=4 \rightarrow a:=0$
R2: $d \geq 4 \rightarrow c:=2$
R3: $d \leq 1 \rightarrow c:=1$

We can see that the attributes *d* and *e* are external, since their value is not inferred by any of the rules. Let IC1 be an integrity constraint defined as IC1: $a+d \geq 2, c \leq 2 \rightarrow \perp$

We will output the context associated with IC1 in this RBS to find out whether the integrity constraint can be violated in any case. According to the first process specification, C(IC1) = Comb(C(a+d≥2), C(c≤2)).

Outputting C(a+d≥2)

In order to output C(a+d≥2), we first need to calculate C(a) and C(d) (see step 1 in the second process). However, the calculation of C(d) is trivial, because d is an external attribute (see step 1.3 in the second process). The rule R1 changes the value of attribute *a*, so this rule is selected. Hence, it is necessary to calculate the context of this rule C(R1):

C(R1) = Comb(C(e≥1), C(d=4)) (third process, step 1.2)

Since the attributes e and d are external, then: C(e≥1) = { (e≥1, ∅) } and C(d=4) = { (d=4, ∅) }, so C(R1) = { ({e≥1, d=4}, ∅) }

In order to output the context C(a+d≥2), we still need to calculate Subst(a+d≥2, [R1]) (step 3.1 in the second process):

Subst(a+d≥2, [R1]) = (d≥2) (step 3.1 in the second process)

Then, C(a+d≥2) = { ({e≥1, d=4, d≥2}, [R1]) } = { ({e≥1, d=4}, [R1]) } (second process)

As the constraint d≥2 is redundant, it can be removed from the environment.

Outputting C(c≤2)

In order to output the context C(c≤2), we need to calculate C(c). There are two rules by means of which the attribute c can be changed, R2 and R3. However, if the rule R3 was selected, the context C(IC1) would later be: C(IC1) = {({....., d=4, d≤1}, [R1;R2])}, which contains an unfeasible set of constraints. Therefore, C(IC1) = ∅. Hence, we only consider the rule R2 for the context C(c). So, C(c) is { ({d≥4}, [R2]) }. Then, we need to calculate Subst(c≤2, [R2]) to output C(c≤2):

Subst(c≤2, [R2]) = (2≤2) (step 3.1 in the second process)

Thus, C(c≤2)= { ({d≥4, 2≤2}, [R2]) } = { ({d≥4}, [R2]) }

And, finally, C(IC1) = Comb(C(a+d≥2), C(c≤2)) = { ({e≥1, d=4, d≥4}, [R1;R2]) } =
= { ({e≥1, d=4}, [R1;R2]) } (first process)

This can be interpreted as: “all the initial FBs in which e≥1 and d=4 will violate the constraint IC1, after firing the rules R2;R1”.

5. Limitations of the Method

We focus on real domain constraints with inequalities in this paper. However, there are other domains like the integer, Boolean and finite domains, etc. [6 p.510-516]. These domains are ignored in our examples, because the algorithms for testing satisfiability in these domains are NP-complete. The Simplex Method, used in this paper to test satisfiability for real domain constraints with inequalities, is exponential time worst case complexity. Nevertheless, it has been observed empirically that Simplex hardly ever needs more than 3m iterations, where m is the number of equations [2]. Hence, the complexity for Simplex can be assumed to be cubic.

6. Conclusion

The two most noteworthy and innovative points of this method are, first, that it can deal with production rules that include some arithmetic constraints on attribute values and, second, that it is a step forward in the verification of non-monotonic RBS. For the purpose of verification, an ATMS-like theory is built as in COVADIS, KB-REDUCER or COVER. However, unlike these methods, our method is able to operate with sets of arithmetic constraints, introducing some new concepts and

procedures taken from other branches of knowledge (Simplex Method, Finite Difference Equation, etc.).

Some problems still remain to be solved. For instance, it will be necessary to improve the efficiency of some procedures used by the method. Our approach to improving efficiency involved using incremental algorithms [6 p.532-533] whenever possible. Moreover, our aim is to integrate this method into another with a wider scope, called MECORI.

References

1. J. Clemente: "*CRIB: Una herramienta para comprobación de restricciones en Sistemas Basados en el conocimiento*". Final-year project, Facultad de Informática, Universidad Politécnica de Madrid, 1994.
2. M. S. Bazaraa, J.J. Jarvis, H. D. Sherali: "*Linear Programming and Network Flows*". Ed. John Wiley Sons.
3. J. Cohen: "*Constraint Logic Programming Languages*". Communications of the ACM. Vol. 33, No.7. pp. 52-68.
4. J. De Kleer: "*An Assumption Based TMS*". Artificial Intelligence 28, 1986.
5. A. Ginsberg. "*Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency and Redundancy*". Proc. AAAI-88, pp. 585-589.
6. J. Jaffar, M. J. Maher: "*Constraint Logic Programming: A Survey*". The Journal of Logic Programming 1994: 19, 20: pp. 503-581.
7. A.D. Preece: "*Verification of rule-based expert systems in wide domains*". Research and Development in Expert Systems VI (Proc. Expert Systems 89), N. Shadbolt, Ed. Cambridge University Press, 1989, pp. 66-77.
8. J. Ramírez: "*Diseño de una Herramienta para la Verificación Semántica de Bases de Conocimiento por Restricciones de Integridad*". Final-year project, Facultad de Informática, Universidad Politécnica de Madrid, 1996.
9. J. Ramírez, A. de Antonio: "*MECORI: a Method for Knowledge Base Semantic Verification Based on Integrity Constraints*". Proceedings of the V&V Workshop at KR'98.
10. M. Rousset: "*On the Consistency of Knowledge Bases: The COVADIS System*". Proceedings ECAI-88, Munich, Alemania, pp. 79-84.

View Selection in OLAP Environment

Shi Guang Qiu and Tok Wang Ling

School of Computing
National University of Singapore
Lower Kent Ridge Road, Singapore 119260
{qiushigu,lingtw}@comp.nus.edu.sg

Abstract. To select some "valuable" views for materialization is an essential challenge in OLAP system design. Several techniques proposed previously are not very scalable for systems with a large number of dimensional attributes in the very dynamic OLAP environment. In this paper, we propose two filtering methods. Our first method, the functional dependency filter, removes views with redundant summary information based on functional dependencies among the dimensional attributes. The second method, the size filter, is based on the view size to filter out any view that can be either derived from another small materialized view or has almost the same number of tuples as another materialized view from which it can be derived. More over, all useful views are selected by these two view filtering methods, other existing view selection methods can still be applied on the remaining views to further reduce other possible non-essential views from systems. We conduct performance tests to compare our method with other existing methods. The results show our method outperform the others.

1. Introduction

On-Line Analytical Processing (OLAP) system is a query subsystem inside a Decision Support System (DSS). It is designed to help users to gain insight into data through fast, consistent, interactive access to a variety of data in the database.

To achieve that, Multi-Dimensional model (MD model) is widely adopted by almost all OLAP systems. Under such model, data attributes are classified according to users' intuitive perception of the business, data are put into a simple and standardized data schema and summary-views**,** some sort of intermediate results, are computed on top of that. User queries are redirected to the smaller summary-views instead of the original sources and better response time can be expected.

On the other hand, the number of possible summary-views in the MD model increases explosively with the increasing number of dimensional attributes. The high storage cost and computation cost make it unfeasible for any system to materialize all of these possible views [OLAP]. Which summary-views should be materialized becomes an essential challenge in OLAP research. Although, there are already quite a number of OLAP products, there are still no good solutions for this problem in the market yet.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 447–456, 2000.

Several methods have been proposed to select an appropriate subset of views or indexes for materialization based on a set of user queries. However, it is very difficult to get an accurate set of user queries, at least at the system design stage. If all possible queries were included, the number of view needs to be considered would increase significantly, the complexity of these methods would be increased.

In this paper, we will approach the problem from the other angle. Instead of finding which view is more useful for OLAP system, we propose two methods to filter out views that are not very useful for the OLAP query optimization process.

Based on functional dependencies among dimensional attributes, we propose a filtering method, **functional dependency filter,** to trim out views holding duplicate summary information. As functional dependencies are normally available at the system design stage, the functional dependency filter can get ride of many views at this initial stage. Functional dependencies are independent on the actual data, so we do not need to review the filtered views unless there are changes to the set of functional dependencies. Also in this paper, we solve the problem for computing functional dependency filter set for a set of functional dependencies.

Then we move our interest to the view size. In [BPT97], a simple method is proposed to filter out those "big" views, which are almost the same size of the ancestor views. In this paper, we extend this technology to "small" views. With today's technology, computers can easily handle complex queries on a view if its size is smaller enough to be kept in the main memory. It is not very useful for a system to materialize any view that can be derived from another already materialized small view.

In these two methods, all useful views are selected. We can still apply other existing view selection methods easily to further reduce those non-essential views from the remaining views selected by these two methods. In the real environment, these two methods are very efficient and a large percentage of views can be filtered out, we might able to directly materialize all the views in this greatly reduced view set.

In section 2, we define summary view and summary view lattice. In section 3, we propose two filtering algorithms and show the experimental results. Finally, the conclusion is presented in section 4.

2. Multi-Dimensional Model and MD Query Model

Before we begin our discussion on view selection, we could like to define some major concepts of MD model and develop some notations and definitions.

2.1 Summary-View and Summary-View Lattice

Multi-Dimensional Model (MD model) [KIM97] is a "business oriented" model. Under such model, all attributes are divided into two major groups, **measure attributes** and **dimensional attributes,** according to their roles in the business analysis. Measure attributes are numeric measurements about the business processes, while dimensional attributes are describing dimensions of the business processes. In an

OLAP SQL queries, measure attributes are normally acting as input for summary operations such as SUM, while dimensional attributes are appearing in Group-by clause and Where clause [KIM97].

Also in a MD model, we can predict queries that users tend to ask and build some intermediate data sets in advance. Based on that, various kinds precomputation methods, e.g. summary-view, range sum cube [HAMS97] and bit-map index [OQ97] have been proposed. Query performance can be improved drastically as we can get the final result by using these smaller precomputed results. Among all, summary-view approach is one of the most popular ones. In the following part of this paper, we concentrate on summary-view only.

Definition 1. Summary-View. A summary-view is an aggregate view grouping some measure attributes along various dimensions, i.e. corresponding to different sets of group-by attributes. In a MD model M, let **GA** be a subset of dimensional attributes, $\mathbf{M_1}$, $\mathbf{M_2}$ .. be measure attributes, $\mathbf{OP_1}$, $\mathbf{OP_2}$ .. be SQL aggregate functions like SUM, COUNT , MAX, and MIN[1]. A summary-view in M is of the form:

SELECT GA, $OP_1(M_1)$, $OP_2(M_2)$, …
FROM relations in M
WHERE (joins of relations in M)
GROUP BY GA

To simplify the discussion, without loss generality, we assume there is only one measure attribute in the M and SUM is the only SQL aggregate function used in the following part of this paper. Thus, we can denote this summary-view as **SV(GA)**.

To represent the relationship among summary-views in a MD model, we will use a lattice framework to represent all possible summary-views in a MD model as a Summary-View Lattice [HRU96].

Definition 2. Operator $\preceq$. We define the operator $\preceq$ between summary-views as below: SV(A) $\preceq$ SV(B) iff SV(A) can be derived from SV(B).

By definition of the summary-view, if A$\subseteq$ B where A and B are two sets of dimensional attributes in a MD model**,** then SV(A) $\preceq$ SV(B), e.g. SV({FAMILY_ID}) $\preceq$ SV({FAMILY_ID,DATE_ID}).

Operator $\preceq$ is a partial order relationship among summary-views, therefore all possible summary-views of a given MD **M** can be presented as a lattice with the summary-view holding all dimensional attributes in its group-by dimensional attribute set as the top element (top view), and the summary-view that aggregates everything together, i.e. nothing in its group-by dimensional attribute set (empty set) as the bottom element.

---

[1] SQL function AVG can be derived from SUM()/COUNT()

Example 1. Let a MD model M which has three dimensioanl attributes A,B,C. The summary-view lattice for M is shown in Figure 1.

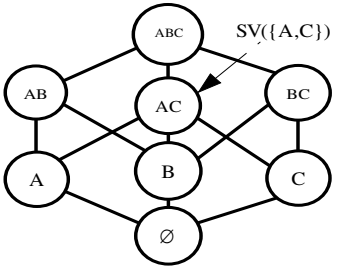


Fig. 1 A Summary-View Lattice

2.2 View Filtering Example

It is obvious that not all these summary-views in summary-view lattice are useful for OLAP query optimization.

Example 2. Assuming we have a simple MD model: sales_info. In it, there are four dimensional attributes: Product, Product_type, Branch_id, Date, one measure attribute Unit, and a functional dependency f: Product→Product_type. Let us look at three possible summary-views in the summary-view lattice:

sv1= SV(Product, Product_type, Branch_id, Date)
sv2= SV(Product, Branch_id, Date)
sv3= SV(Product_type, Branch_id, Date)

First, we look at sv1 and sv2. As every Product belongs to only one Product_type, two summary-views have the same number of tuples and hold the same summary information. Only one should be considered for materialization. In this paper, sv1 will be chosen as it can be used to compute all queries answered by sv2.

Note that how sv1 should be materialized will be considered as a separate issue. Whether sv1 should be materialized directly as a flat table or split into sv2 and a small table p=(Product, Product_type), and how these views should be indexed will not be covered in this paper. Here, we concentrate on eliminating redundant summary information for measure attributes, this is, sv1 and sv2 should never be selected at the same time.

Next, we look at sv3, should it be materialized? If sv1 is small, e.g. whole sv1 can be fetched into memory quickly, it may not worth to materialize sv3 separately. All queries running against sv3 can be redirected to sv1, additional I/O cost is negligible. In another case, if there are only few Product belongs to each Product_type, sv1 and sv3 might have almost the same number of tuples. It might be not worth to materialize sv3 also. The benefit gained in query processing might not justify for the storage cost involved.

In the above example, we have made use of two view trimming methods, which can potentially trim out a large percentage of summary-views from a summary-view lattice. In the next sections, we propose these two filtering algorithms formally.

3. Summary-View Lattice Trimming

The total number of possible **summy-view** for a MD model with n dimensional attributes will be 2^n, it is simply impractical to materialize all nodes inside the lattice. We should only materialize those views of the most value for the system performance.

3.1 Related works

Most techniques proposed previously concentrate on selecting an appropriate subset of views or indexes based on a given set of user query. Various kinds of heuristics, e.g. greedy, A* algorithm, are used to obtain a near optimal solution [HRU96][GHRU97] [G97][YKL97][TS98][BPT97].

In [HRU96], the overall computation cost of the given set queries inside the lattice is considered. By using the Greedy algorithm, the most "valuable" view for the remaining unmaterialized views is picked at each time, until the stop condition is met. In [BPT97], a size filter algorithm is proposed to filter out those summary views with similar number of tuples as their ancestor views.

In the very dynamic OLAP environment, these methods are hard to be implemented.

1) It is very difficult to make good predictions for user queries, especially at the system design stage. If all possible queries were included, the number of possible views will be increased significantly.
2) It is also very difficult to evaluate the size of the views accurately, which are used to estimate the view costs. Before the actual data set is populated, available information is very limited, especially for those expensive big views. After the data is loaded, we still need to do some complex statistics information collection. One of the major burdens is the large number of view need to be evaluated. Sampling methods, e.g. [DNK+97], could be a choice, but its accuracy is depending on the samples. More, data pattern in the OLAP environment is also changing frequently. We need to evaluate view sizes repeatedly.
3) In addition, there is no simple way to verify whether enough views have been selected.
4) At last, various kinds of view selection algorithms are not compatibility with each other. Some useful views selected by one algorithm could be filtered out by another based on different view selection criteria without compensation. It is difficult to use more than one method in the same case.

3.2 Functional Dependency Filter

Definition 3. Functional Dependency Filter, $\varphi(f)$

Let M be a MD model, $f:AL \rightarrow AR$ be a functional dependency where AL and AR are two subsets of dimensional attributes. The functional dependency filter for f, denoted as $\varphi(f)$, is defined as: $\varphi(f)=\{SV(X) \mid AL \subseteq X \wedge AR \not\subset X\}$ where X is a subset of dimensional attributes.

Theorem 1. For any view SV(X) in $\varphi(f:AL \rightarrow AR)$, we can find another summary-view $SV(X \cup AR)$, which is in the summary-view lattice but not in $\varphi(f)$ and holds the same summary information as SV(X). Therefore, all views in $\varphi(f)$ should be filtered out

Proof is included in [QSG].

Definition 4. Functional Dependency Filter Set.

Let $F=\{f_1,f_2,\ldots f_n\}$ be a functional dependency set in a MD model M where each f_i is a functional dependency on dimensional attributes. We define the functional dependency filter set for F, denoted as $\varphi(F)$, as a union of $\varphi(f_i)$ where i=1 to n

Example 3. Assuming there is a MD model M with four dimensional attributes {A, B, C, D} and $F=\{f_1:A\rightarrow B, f_2:BC\rightarrow D\}$. We can easily get $\varphi(F) = \varphi(f_1)\cup\varphi(f_2)$ = {SV({A}), SV({A,C,D}),SV({A,C}),SV({A,D}),SV({A,B,C}),SV({B,C})} as shown in Fig. 2. Views in $\varphi(F)$ are represented as shaded nodes. Views in bracket below the node are holding the same summary information and replaced by it, e.g. SV({A,B}) is used to replace SV({A}). It is also interesting to look at that derived functional dependency $f_3:AC\rightarrow D$, $\varphi(f_3)$={SV({A,C}),SV({A,B,C})}, $\varphi(f_3)$ is a subset of $\varphi(F)$.

As a next step, we compute a functional dependency filter set for the closure of a set of functional dependencies. There are two problems: completeness and Interference. As there are many derived functional dependencies, will the functional dependency filters for derived functional dependencies further filter out any new summary views? Will the filter sets of functional dependencies in the closure interfere with each other and filter out some useful views?

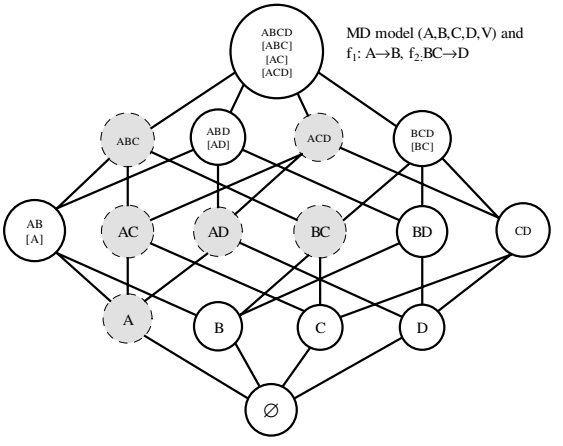


Figure 2: Functional Dependency filter

Theorem 2. For a set of functional dependencies F in M, $\varphi(F)= \varphi(F^+)$ where F^+ is the closure of F.

Proof will be included in [QSG].

Based on the Theorem 2, $\varphi(F)$ has already included all summary-views need to be filtered out in $\varphi(F^+)$ and $\varphi(F)$ is complete. We don't have to worry about the functional dependency filters for functional dependencies derived from F.

If a view SV(X) is filtered out by functional dependency filter $\varphi(f_1:AL_1\rightarrow AR_1)$, then $SV(X\cup AR_1)$, which is not filtered by $\varphi(f_1)$, holds the same summary information as SV(X) , by Theorem 1. Similar, if $SV(X\cup AR_1)$ is filtered out by another functional

dependency filter $\varphi(f_2:AL_2 \rightarrow AR_2)$, then $SV(X \cup AR_1 \cup AR_2)$, which is not filtered by either $\varphi(f_1)$ or $\varphi(f_2)$, holds the same summary information as $SV(X \cup AR_1)$ and $SV(X)$. Similarly, we can prove that for every view filtered out by functional dependency filters, there must be one view remained in the summary-view lattice which holds the same summary information for measure attributes. So we don't have to worry interference among the filter sets for all functional dependencies also.

Hence, we can compute the functional dependency filter set for the functional dependency closure by union the functional dependency filter for each functional dependency inside F at any sequence. Also, assuming F' is a non-redundant covering of F. As $F^+=F'^+$, so $\varphi(F)= \varphi(F^+)=\varphi(F')$. Cost to compute $\varphi(F)$ could be further reduced by computing $\varphi(F')$ instead.

As functional dependencies are normally available at the system design stage, functional dependency filter can filter out a large number of views in advance before actual data are loaded. Thus, system initialization, system maintenance, and further view selection tasks can be drastically simplified. In addition, we do not have to review the filtered view unless there are changes in the set of functional dependencies.

3.3 Size Filter

Not all summary-views are useful for OLAP query optimization. Let the size of a materialized view V be a function: SIZE(V). When SIZE(V) is smaller than a certain value, such as total memory available for the application, it is not a big issue for the OLAP system to process complex queries posted on V within the satisfied response time. So, there is no need to materialize any other summary-view that can be derived from this small view. The extra CPU, I/O cost in query processing to use this slightly bigger view is tolerable. In this paper, we will call the threshold as **Lower_Bound.** In Figure 3, if SV({A,B,C}) is very small and has been materialized, we can directly filter out all tiny views under it in the lattice.

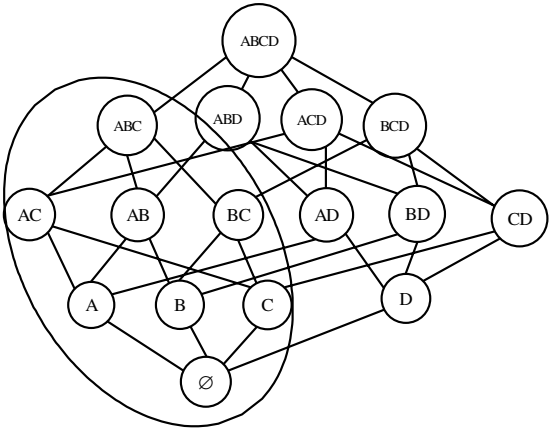


Fig. 3 Size Filter of Summary-View Lattice

Because of high aggregation, a view with few dimensional attributes in its group-by attribute set will hold similar number of tuples as its domain size, the number of possible distinct tuples in the view. As most of the dimensional attributes have very small

domains, e.g. either male or female for human gender, a large number of small views can be filtered out. The OLAP System can be drastically simplified, although the storage space saved might not very significant.

In another case as it described in [BPT97], if a view Y derived from another materialized view X has almost the same number of tuples as X, view Y would be similar to a sorted version of view X, there would be little benefit for us to materialize the view Y. In this paper, we define a threshold value as the **Upper_Bound_Ratio**. If SIZE(Y)/SIZE(X) > Upper_Bound_Ratio, we will filter view Y from the summary-view lattice. In an OLAP system with a large number of dimensional attributes, many summary-views, especially those large views near the top of the summary lattice, could be filtered out by this method.

In the size filtering method, the views need to be processed strictly from the top to the bottom of the summary-view lattice in order to avoid the conflict among different steps of the size filter algorithm.

In the OLAP environment, it is difficult to estimate the view size accurately. However, our size filter is running after applying functional dependency filters, much smaller number of views need to be processed.

Size filter is to achieve a tradeoff between performance and cost. Upper_Bound_Ratio and Lower_Bound are determined based on various factors. Beside machine capabilities, e.g. CPU, Memory, harddisk space and system I/O speed, maintenance cost and user requirement are also very important. For example, we could define a bigger Lower_Bound if queries are asked in the system are simple or a faster harddisk array is available. In the actual system, size filter can still get ride of many views in the summary lattice even with conservatively selected bound values, e.g. 95% as the Upper_Bound_Ratio and 10MB as the Lower_Bound.

3.4 Experiments

In this test, data and queries are generated by the program downloaded from the website [OLAP][2]. The system used is a Pentium Celeron 366 with 128MB SDRAM, running Windows NT 4.0 and Oracle 8.04. In this example, we use only one fact table, four dimension tables and eight non-redundant functional dependencies among dimensional attributes (Figure 4). All these data in form of raw text files are about 80MB.

In the test, we assume all summary-views will be used equally. Three view sets are built.

1) The first view set is generated by our algorithms. We apply the functional dependency filtering algorithm on this MD model using 8 FDs and filter out 3680 summary-views out of 4096, or 95.89%. For the size filter, we use 80% as the Upper_Bound_Ratio and 10MB as the Lower_Bound, filter out another 135 views and get 33 views[3].

---

[2] The parameters used as input to the data generator program are: Channel: 10, Density: 0.1% and Users: 10. Refer the [OLAP] for more detail about the data generator.

[3] To simplify the size estimation problem, we actually count the number of tuples in these 168 views, and assume: VIEW SIZE = Records Length × Number of tuples.

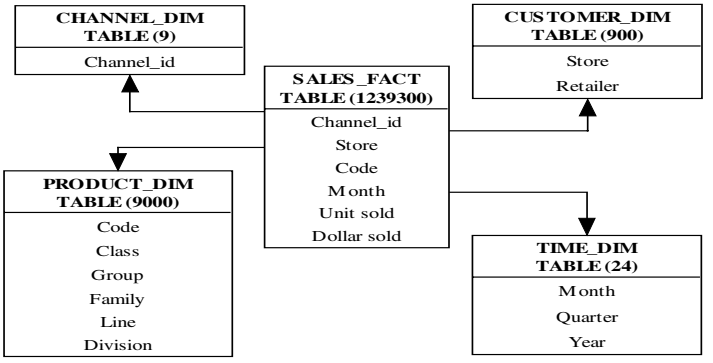


Fig. 4 Multidimensional model
The number on the right of the table name indicates the number of tuples in that table
FD: Code→Class, Class→Group, Group→Family, Family→Line, Line→Division
Store→Retailer, Month→Quarter, Quarter→Year
*(Year is include as an element in Month, Quarter. e.g. 021997 and Q21997)

2) The second set is generated by algorithm proposed by [BPT97] running on the 168 views selected by our functional dependency filter.[4] We use 80% as the Upper_Bound_Ratio, get 135 views.
3) The last data set is generated by [HRU96] with Maximum 33 views as the stop condition for the greedy algorithm.

For the benchmark test queries, we use Query Set 1 (Channel Sales Analysis) generated by the data generator program [OLAP] also. There are 2500 queries in this set, we only use the first 500 queries because of the resource limitation. Test results are shown in Table 1.

Methods	Views selected	Space required	Processing Time (avg)	Queries processed within 3 sec		Query processed within 3 ~ 20 sec.		Queries processed more than 20 sec	
				No. of Queries	Processing Time (avg)	No of Queries	Processing Time (avg)	No. of Queries	Processing Time (avg.)
Ours	33	1.09GB	9.1 sec.	312	1.16 sec.	35	9.67 sec	153	25.24 sec
[BPT97]*	135	1.16GB	9.1 sec.	312	1.16 sec.	35	9.67 sec	153	25.24 sec
[HRU96]	33	1.93GB	8.9 sec.	311	1.22 sec	53	12.28 sec	136	25.05 sec

Table 1. Performance Comparison
* [BPT97] is running after applying Functional Dependency Filter

It is clearly that our approach is the overall winner. We have reduced the number of summary-views in the summary-view lattice to 1% of the original size. To achieve a similar performance, we use the same number of views with 56% of storage space required by [HRU96] and only 25% of views required by [BPT97][5]. More testing cases explored in [QSG] shows the similar results.

[4] It is because [BPT97] yields very bad result if it is simply applied on all possible views and also we want to compare the effort of Lower Bound Size Filter.
[5] [BPT97] is running after applying our functional dependency filter

4. Conclusion and Future Works

In this paper, we are proposing functional dependency filter and a size filter to filter out a large number of redundant views in the OLAP system with limited cost of query performance. In the test, our algorithms generated an impressive result comparing with those of others.

As one of the future work, we would like to study how to accelerate query which has many dimensional attributes in its where clause and few dimensional attributes in its group-by clause. Summary-view approach may not be a good solution, as heavy aggregations are required to run against a big summary-view. To organize this kind of big views for quick access is still an open problem. Complex indexes [OQ97], Range-Cube [HAMS97], sub-cube and some other complex data structure could be the right direction. Further research in this area is definitely needed.

Reference:

[BPT97] E.Baralis, S.Paraboschi, E.Teniente. Materialized View selection in a Multidimensional Database, VLDB'97

[DNR+97] P.M.Deshpande, J.F.Naughton, K.Ramasamy, A.Shukla, K.Tufte, Y.Zhao Cubing Algorithms, Storage Estimation, and Storage and Processing Alternatives for OLAP, Bulletin of Technical Committee on Data Engineering Mar 1997, pp. 3 – 11

[G97] H.Gupta, Selection of Views to Materialize in a Data Warehouse, Proc. ICDT'97

[GHRU97] H.Gupta, V.Harinarayan, A.Rajaraman, and J.D.Ullman. Index selection for OLAP. In Proc. ICDE'97, pp. 208 – 219

[HAMS97] C.Ho, R.Agrawal, N.Megiddo, R.Srikant, Range Queries in OLAP Data Cubes SIGMOD'97, pp. 73-88

[HRU96] V.Harinarayan, A.Rajaraman, J.D.Ullman, Implementing Data Cubes Efficiently, ACM SIGMOD '96, pp. 205-216

[KIM97] R.Kimbal A Dimensional Modeling Manifesto AUG 1997 //www.dbmsmag.com

[OQ97] P.O'Neil, D.Quass, Improved Query Performance with Variant Indexes, SIGMOD'97

[OLAP] //www.olapcouncil.org

[QSG] S.G.Qiu, View Selection in OLAP Environment, Thesis for master degree, NUS

[TS98] D.Theodoratos, T.Sellis, Data Warehouse Schema and Instance Design, ER'98

[YKL97] J.Yang, K.Karlapalem, Q.Li, Algorithms for materialized view design in data warehousing environment, Proc. VLDB' 97, pp. 136-145

Storage Management of a Historical Web Warehousing System*

Yinyan Cao[1], Ee-Peng Lim[2], and Wee-Keong Ng[3]

Centre for Advanced Information Systems, School of Applied Science, Nanyang Technological University, Nanyang Avenue, Singapore 639798, SINGAPORE,
aseplim@ntu.edu.sg

Abstract. In this paper, we present the storage management of the WHOWEDA web warehousing system, which warehouses historical web information. To facilitate inter-table and intra-table sharing of web pages, we propose a three-layer storage architecture, that consists of tuple, table, and pool layers of storage modules storing different parts of warehoused web information. To improve retrieval efficiency, we have chosen to replicate some node attributes across web tables in the table layer while keeping only unique copies of web pages at the pool layer. The separation of table and pool layer storage also allows different valid times to be maintained by multiple web tables for the same web pages due to different schedules of global coupling across web tables. As the sharing of web pages may lead to valid time inconsistency between different web tables, we propose an update synchronization scheme to resolve the valid time differences on user request.

1 Introduction

The World Wide Web (the Web) changes autonomously. In a survey done by Internet Archive, the average life span of web pages is only 75 days [1]. As web sites update their web pages regularly, Internet users can easily obtain the most up-to-date information using web browsers. Nevertheless, once a web page is updated, its content is overwritten. Its previous content can never be retrieved again from the web site unless the web site administrator provides some facility to archive web pages and make the archived information available to the web users.

To manage historical web pages and to systematically derive useful knowledge from the Web, we are developing a web warehousing system known as **WHOWEDA** (WareHouse Of WEb DAta)[17]. It stores and manages historical web information extracted from selected web sites [7]. WHOWEDA adopts a temporal web data model, which represents web information as a set of **web**

* This work was supported in part by the Nanyang Technological University, Ministry of Education (Singapore) under Academic Research Fund #4-12034-5060, #4-12034-3012, #4-12034-6022. Any opinions, findings, and recommendations in this paper are those of the authors and do not reflect the views of the funding agencies.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 457–466, 2000.

tables. Each web table consists of a set of **tuples** described by a **schema**, which defines constraints on the structure and content of the tuples. An example of a web table is *News* shown in Figure 1. This web table models some segments from the Business Times web site. It contains three tuples and each tuple is a directed graph of **nodes** and **links**. A node represents a web page, while a link represents a hyperlink. The **attributes** of a node include *url*, *title*, *format*, *size*, *text*, and *valid time*. The valid time is an interval enclosed by two time points *last-modified time* and *downloading time*, which refer to the time points at which the web page is updated or created and when the update is recorded into the warehouse respectively. We obtain their values from Last-modified time and Date in HTTP message headers. In this paper, we use *last-modified time* and *downloading time* for a node object, and Last-modified Time (LMT) and Downloading Time (DT) for a web page. A web schema consists of a set of **node variables**, a set of **link variables**, a set of **connectivities**, and a set of **predicates**. Variables, and connectivities determine how the node and link instances are connected in the tuples belonging to the web table. The node and link variables further allow us to specify predicates on the attributes of node and link objects.

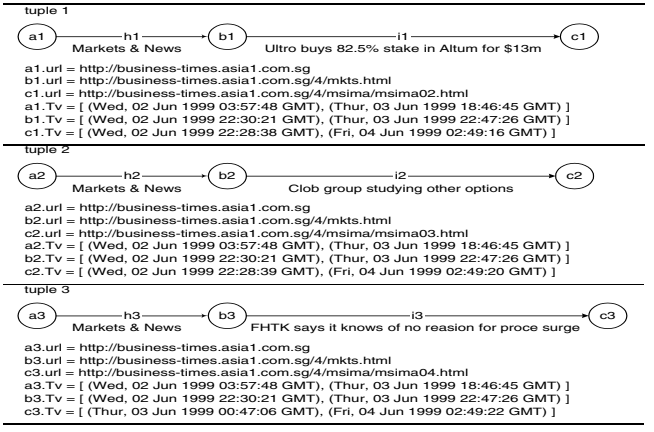


Fig. 1. Web Table *News*

In order to create and manipulate temporal web tables, we have defined several web operators (Refer to [7] for detail). In particular **global coupling** operator retrieves inter-connected web documents from the Web. The web tables directly captured from the Web are called **base tables**. Further data manipulation and analysis can be performed on the existing web tables by a set of local web operators. Web tables generated by these operations are known as **derived tables**.

In this paper, we address the issues and design choices of storage management in WHOWEDA. To illustrate the various issues and design choices, we use three base web tables as examples. The first web table *News* (see Figure 1).

The second web table *Reports* (see Figure 2) captures from the same site some tuples about regional market reports and their stock exchange tables. The third web table *Fish* captures from the web site http://www.fish.com.sg[1] some tuples about the prices of stocks listed in Singapore Stock Exchange. The storage management of WHOWEDA is more than just archiving web pages, because each web page is associated with some semantic meaning that needs to be modeled and stored. The design of the WHOWEDA's storage management must consider the following issues:

- **Storage Scalability:** Since WHOWEDA warehouses historical web information, the number of warehoused web documents is large and always growing. More and larger base tables will be created as new tuples are harvested from the Web over time. More derived tables will also be generated as they are defined by users for different applications. For example, the daily update in the three example base web tables requires about 10MB of storage space to store new web pages. Thus, scalability in storage is critical.
- **Update Synchronization:** To maintain historical information from the Web, web tables require periodical global coupling. The global coupling operation introduces new tuples, nodes, links, and web pages into the web tables. It also updates the valid time of some existing web pages. Upon user request, the update on the valid time of these web pages should be propagated to the other web tables sharing these web pages.
- **Performance Scalability:** The insertion and retrieval cost should remain reasonable, while the storage cost is optimized.

The above storage management issues are by no means exhaustive. For a start, we have limited our study to the following scope: (1) we assume there is one large capacity hard disk with sufficient storage space for warehousing, (2) we consider standard HTML pages only since their content are easier to query compared to other multimedia objects. (3) we have not supported the deletion of nodes, links, and tuples from web tables as we assume the main purpose of the warehouse is for querying, (4) the archival facilities have not been included, and (5) we have not included concurrency control, recovery, and caching mechanisms. Further extensions of our proposed storage management scheme will consider them.

The outline of this paper is as follows. A three-layer storage architecture will be presented in Section 2. We consider the other storage issues in Section 3. The related work is described in Section 4. We summarize our work and provide future directions in Section 5.

2 Intra and Inter-table Sharing

In WHOWEDA, two types of object sharing are identified, namely **intra-table sharing** and **inter-table sharing**. Given a web table, one can often find different tuples consisting of nodes having the same *url* and *last-modified time* for the

[1] The site displays real time stock information.

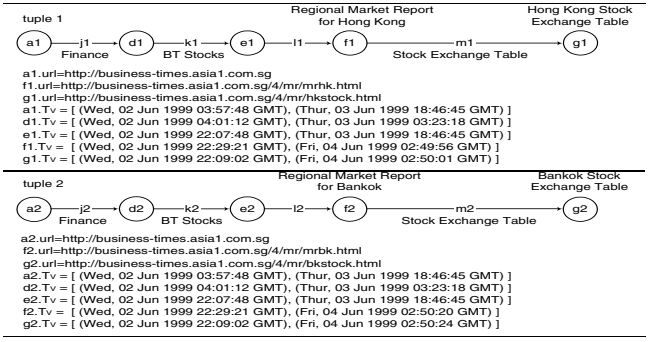


Fig. 2. Web Table *Reports*

same *node variable*. This intra-table sharing of common node information arises when the tuples are extracted from the same web sites using global coupling. For example, in web table *News* shown in Figure 1, nodes a_1, a_2 and a_3 refer to the same web page[2]. All three nodes are instances of the node variable a. Similarly, the link objects from different tuples may share the same information. For example, in *News*, links h_1, h_2, and h_3 refer to the same hyperlink, which is from web page of a_1 to web page of b_1. All of them are instances of the link variable h.

Different web tables may share some common web documents, each is uniquely identified by its URL and LMT. For example, node a_1 in *News* and node a_2 *Reports* (as shown in Figure 2) correspond to the same web page[3]. This is known as inter-table sharing.

2.1 Three-layer Storage Architecture

A three-layer storage architecture has been adopted to facilitate the intra and inter-table sharing. The overall three-layer architecture is illustrated in Figure 3. The tuple layer storage maintains the tuples for different web tables. The table layer storage consists of **table nodes** and **table links**, that correspond to all nodes and links. The pool layer storage consists of a set of **pool nodes**, each of which is associated with a unique web document maintained by WHOWEDA. The tuple and table layer storage are related by intra-table sharing. The table nodes and links are shared by tuples from the same web table. The table and pool layer storage are related by inter-table sharing. The pool nodes are shared by table nodes across different web tables.

In Figure 3, there are three segmented rectangles spanning across the tuple layer and the table layer. Each of these rectangles outlines a web table. The tuples from a web table use only table nodes and table links from the web table.

[2] URL=http://business-times.asia1.com.sg, LMT=(Wed, 02 Jun 1999 03:57:48GMT)

[3] URL=http://business-times.asia1.com.sg/4/mkts.html, LMT=(Wed, 02 Jun 1999 22:30:21 GMT)

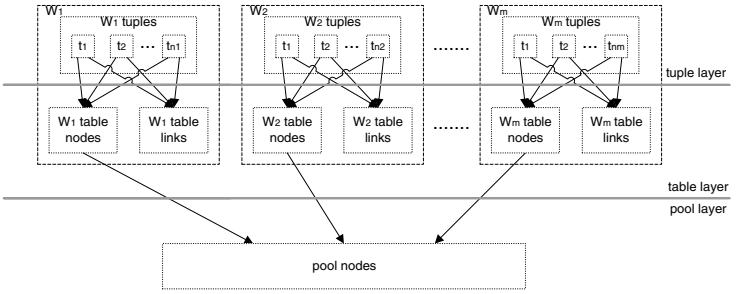


Fig. 3. Three-layer Storage Architecture

However, table nodes from different web tables have to share a common set of pool nodes.

Figure 4 shows the sharing between tuple layer and table layer within one web table. A table node contains the information shared by nodes from different tuples. For example, nodes a_1, a_2 and a_t all have the same *url*, *last-modified time* and *node variable* and they share table node 1. Each table node is assigned an identifier unique in its web table. Similarly, a table link contains the information shared by links from different tuples. Two links share the same table link, if their source nodes share the same table node and their target nodes share the same table node. Each table link is assigned an identifier unique in its web table. A tuple is represented as a set of connected identifiers of table nodes and table links. Each tuple is assigned an identifier unique in its web table.

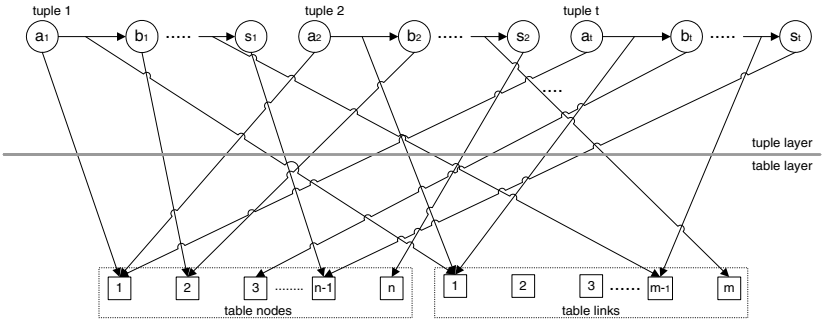


Fig. 4. Intra-table Sharing

Figure 5 shows the sharing between table and pool layer. It does not show tables links since inter-table sharing is reflected only in the relationship between table nodes and pool nodes. A pool node is associated with a unique web document collected by WHOWEDA. Each pool node is assigned an identifier unique in the pool layer. A table node keeps the identifier of the pool node if they

capture the same web document. The information of a node is distributed physically among its corresponding table node and pool node, which is described in Section 3.1.

2.2 Cost Analysis

The separation of table layer from tuple layer has two advantages: (1) it saves storage space especially when intra-table sharing is heavy, (2) the predicates in our web data model specify constraints directly on nodes and links, so keeping them separate from tuples helps in the direct access of nodes and links and the evaluation of predicates. Assuming the pool layer stores all unique web documents harvested by WHOWEDA, we compare the cost of storing a web table in tuple and table layers with the cost of storing nodes and links physically inside each tuple. We first created one base table *News*, a portion of which is shown in Figure 1. The table consists of 28 days of data and contains 289 tuples on the whole. If the table is stored in tuple and table layers with intra-table sharing, the storage requirement is 74,320 bytes. If nodes and links are physically stored with each tuple except that the text of nodes are stored in pool layer, the storage requirement is 161,024 bytes. Hence, over half of the storage space has been saved with the separation of tuple and table layer.

The separation of pool layer from table layer greatly saves storage space by avoiding duplicates of web pages. Here we investigate the saving associated with pool layer in derivation sharing. Let w_1, w_2, $\cdots$, w_n be a set of web tables, assume that w_1 is a base table, and $\forall i, 1 < i \leq n$, each w_i is the resultant web table of a web operation whose operand is w_{i-1}. Let d_i denote the number of table nodes in w_i. We have $d_i \leq d_{i-1}$ as web operations do not create new nodes in the derivation process. Assume that the average size of a web page is S. If we store the web documents in pool layer, the space occupied by web pages in web tables w_1 to w_n is $d_1 * S$, which is a constant value. If we remove the pool layer and store a web document with each table node, the storage space is $(d_1 + d_2 + \cdots + d_n) * S$. We further assume that the reduction of size between two consecutive tables is defined by a constant factor $0 < q \leq 1$. With pool layer, the total storage space becomes $d_1 * S * (1 - q^n)/(1 - q)$ for $0 < q < 1$ and $d_1 * S * n$ for $q = 1$. Clearly, without pool layer storage, the storage space required for web documents will be very large.

3 Other Storage Management Issues

3.1 Attribute Replication

In the three-layer storage architecture, the information contained in a node is organized into two parts and stored in table node and pool node separately. To optimize the utilization of storage space, information common to all table nodes capturing one web document should be stored in the shared pool node, the information specific to a web table should be stored in the table node. However, if

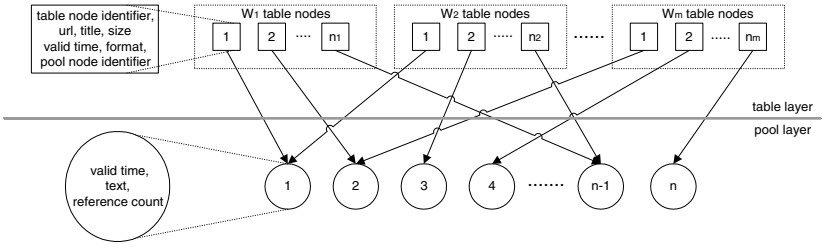


Fig. 5. Inter-table Sharing

these shared information are frequently requested, pool layer access may become a performance bottleneck. Thus, we move some of the shared information from pool nodes to table nodes to reduce the traffic to the pool layer.

Figure 5 illustrates the information distribution among pool nodes and table nodes. Among the six attributes of a node object, *text* should be stored with pool node because it is identical to all corresponding table nodes and its size is large. We decide to store *valid time* in both pool nodes and table nodes for two reasons. Firstly, creation of different web tables deploys different supporting operators which may modify the valid time of nodes of the web table. Therefore, nodes modeling the same document from different tables may have different valid time. On one hand, the original copy of valid time should be kept in pool node to faithfully record the history of web documents. On the other hand, the table nodes can store the valid time assigned at the time the web table was last created or updated. The second reason is that a web document may be updated at any time in its host machine. A web table involving the updated document may capture the update through global coupling. Upon explicit user request, the other web tables involving the document can have the corresponding table nodes updated. Since the pool layer is shared by all web tables, it is convenient to keep the latest valid time in pool node so that all updates by global coupling are tracked in the pool layer. In this case, the pool layer can serve as a relay station to all the updates. A detailed discussion on the synchronization of valid time of table nodes will be given in Section 3.2.

For the remaining attributes, we can allocate them to either table nodes or pool nodes. By keeping them in pool nodes, storage cost is reduced. The pool layer needs to be accessed if any of them is requested, e.g. by query processor. The pool layer access introduces extra processing time. If the access frequency of these attributes is very high, the access to pool layer will become a performance bottleneck. If we replicate the attributes in each corresponding table node, pool layer access is reduced with some increase in storage cost. This is a relatively inexpensive option considering the small attribute sizes and the gain in retrieval efficiency. For example, *size* is an integer, and *title* is a string usually containing only a few words. In our *News* table, the size of the table node file is 48,092 bytes with replication, and 10,665 without replication. The cost of replication

is insignificant compared to the storage space occupied by the web pages, i.e. 4,372,992 bytes.

3.2 Update Synchronization

The web documents in WHOWEDA can be classified into **current** and **history documents**. A current document is one last know to be valid on the Web. A history document is one that once existed on the Web but has been modified. A table node is a **current table node** if it represents a current document. Otherwise, it is a **history table node**. A *status* attribute is assigned to each table node. The status value of a current table node is CURRENT. The status value of a history table node is OBSOLETE. Similarly, there are current and history pool nodes.

In both table layer and pool layer, the history nodes are considered consolidated and not subject to further change, while current nodes are subject to change. According to the datum that is updated, there are three types of updates: namely **valid time update**, **status update**, and **creation update**. Valid time update refers to updating the *downloading time* of a current node to a new value. Status update refers to updating the *status* attribute from CURRENT to OBSOLETE. It implies that the node is turned into a history node from a current node. Creation update refers to the creation of new table node and new pool node for each new web document.

There are two types of update operations: **global coupling** and **explicit synchronization**. The pool layer serves as a central repository capturing all updates by global coupling operations and propagating them to the other web tables in explicit synchronizations. Global coupling is performed periodically to maintain a temporal web table. It retrieves a set of inter-linked web documents from the Web satisfying a given schema. Associated with each document are three most important attributes, namely URL, LMT, and DT. Updates to URL and LMT may cause status update and creation update, while updates to DT may cause valid time update. With each web document retrieved by global coupling, the pool layer generates a key with the URL and LMT to determine whether the web document already exists in the warehouse. For a pre-existing web document, the pool layer compares the *downloading time* of the associated pool node and the new DT of the document. If the *downloading time* is earlier than the DT, valid time update is performed on the pool node so that its *downloading time* is assigned to the new DT. If the web document has not been stored in the pool layer, creation update is performed to create a new pool node for the web document. If there exists a current pool node such that its *url* equals the URL and its *last-modified time* is different from the LMT, status update is performed to turn it into a history pool node. The updates on table nodes in table layer are similar to that of pool layer.

Explicit synchronization is performed upon user's explicit request if the user needs to view or analyse the web table with the latest updates in the warehouse. It updates all current nodes of a web table with the changes introduced by the

other web tables. For each current table node in the web table, the valid time and status are updated to that of the corresponding pool node.

4 Related Work

Our temporal web data model is built upon concepts from diverse research areas, mainly web data models and temporal databases. The previously proposed temporal data models usually extend relational data models [8, 16, 18]. Our data model is designed to overcome limitations of relational model which is flat and unsuitable for Web. A detailed survey on database techniques applied on the Web is given in [10]. WebSQL [4, 15] and W3QL [13] assume the internal structure of document is not known. WebLog [14] is a logic based language for querying as well as restructuring the Web. WebOQL [3], Lorel [2], UnQL [6], and Florid [11] are also designed to query semistructured data. Araneus [5] and Strudel [9] are both web site management systems, supporting restructuring and creation of Web sites. Our work follows the same line as WebSQL in assuming no knowledge on the internal structure of web documents. Different from all the above research, our data model captures historical web data, and web segments are modeled as web tables, which are treated as first-class objects. Similar to Florid, the captured information are stored locally. While the other research works focus on web restructuring besides querying, we concentrate on further manipulation and analysis of web data and to provide the DBMS support for web applications.

Both WebBase [12] and the storage manager of WHOWEDA construct and maintain a large repository of web pages. However, there are some notable differences. WebBase archives large number of arbitrary and relatively fresh web pages without correlating their semantic relationship, since WebBase does not have a data model. Our storage manager archives user interested web segments with schemas describing the semantics of these web pages. Web table is the highest abstraction in our data model. The Internet archive [1] aims to record the whole of the online world and they preserve the data to ensure their longevity. Its purpose is quite different from that of WHOWEDA.

5 Summary and Future Work

We have designed and implemented a storage manager which is equipped with the capability of temporal queries. Inter-table sharing and intra-table sharing have been identified as the main storage issues. In consequence, a three-layer storage architecture has been adopted with tuple and table layer dedicated to storing web tables and pool layer dedicated to storing and manipulating web pages. Issues such as scalability and synchronization have been considered. In the future, we will work in the following three directions: (1) we will investigate how to allocate web documents among multiple disks to support efficient retrieval, (2) we will consider direct stream access to web documents to facilitate applications such as data mining, and (3) we will incorporate archiving facility into the storage manager.

References

1. http://www.archive.org.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.
3. G. Arocena and A. Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proceedings of ICDE'98*, Orlando, Florida, February 1998.
4. G. Arocena, A. Mendelzon, and G. Mihaila. Applications of a web query language. In *Proceedings of the 6th International WWW Conference*, Santa Clara, April 1997.
5. P. Atzeni, G. Mecca, and P. Merialdo. To weave the web. In *Proceedings of the 23rd VLDB Conference*, Athens, Greece, 1997.
6. P. Buneman, S. Davidson, and G. Hillebrand. A querying language and optimization techniques for unstructured data. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 505–516, Montreal, Canada, 1996.
7. Y.Y. Cao, E.P. Lim, and W.K. Ng. On warehousing historical web information. Technical report, Centre for Advanced Information Systems, Nanyang Technological University, Singapore, September 1999.
8. J. Clifford and A. Croker. The historical relational data model (HRDM) and algebra based on lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537. IEEE Computer Society, February 1987.
9. M. Fernandez, D. Florescu, J. Kang, and A. Levy. Catching the boat with Strudel: Experiences with a web-site management system. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Seattle, WA, 1998.
10. D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide web: A survey. *ACM SIGMOD Record*, 27(3):59–74, September 1998.
11. R. Himmeroder, G. Lausen, B. Ludascher, and C. Schlepphorst. On a declarative semantics for web queries. In *Proceedings of the 5th International Conference on Deductive and Object-Oriented Databases*, Montreux, Switzerland, December 1997.
12. J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: a repository of web pages. Technical report, Stanford University, 1999.
13. D. Konopnicki and O. Shmueli. W3QS: A query system for the world wide web. In *Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1995.
14. L. V. S. Lakshmanan, F. Sadri, and L. N. Subramanian. A declarative language for querying and restructuring the web. In *Proceedings of the 6th International Workshop on Research Issues in Data Engineering, RIDE '96*, New Orleans, February 1996.
15. A. Mendelzon, G. Mihaila, and T. Milo. Querying the world wide web. *International Journal on Digital Libraries*, 1(1):54–67, April 1997.
16. S.B. Navathe and R. Ahmed. A temporal relational model and a query language. *Information Sciences*, 49(1-3):147–175, 1989.
17. W.-K. Ng, E.-P. Lim, C.-T Huang, S.S. Bhowmick, and F.-Q. Qin. Web warehousing: An algebra for web information. In *Proceedings of IEEE International Conference on Advances in Digital Libraries (ADL'98)*, April 1998.
18. Richard Snodgrass. The temporal query language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

Range-Max/Min Query in OLAP Data Cube

Hua-Gang Li Tok Wang Ling Sin Yeung Lee

Department of Computer Science
School of Computing
National University of Singapore
Lower Kent Ridge Rd, Singapore 119260
{lihuagan, lingtw, jlee}@comp.nus.edu.sg

Abstract. A range query applies an aggregation operation over all selected cells of an OLAP data cube where the selection is specified by ranges of continuous values for numeric dimensions. Much work has been done with one type of aggregations: SUM. But little work has been done with another type of aggregations: MAX/MIN besides the tree-based algorithm. In this paper, we propose a new method which partitions the given data cube, stores pre-computed max/min over partitions and location of the max/min of the partitions. We also use some techniques to reduce the chance of accessing the original data cube when answering ad hoc queries at run time. The experiment results demonstrate that our method outperforms the tree-based algorithm.

1 Introduction

The data cube [6], also known as the multidimensional database (MDDB) in the On-Line Analytical Processing (OLAP) [4] system, is designed to provide an intuitive way for data analysts to navigate various levels of summary information in the database and data warehouse. In a data cube, attributes are categorized into *dimension attributes* and *measure attributes*. The measure attributes of those records with the same functional attributes values are combined (e.g. summed up) into an aggregate value. Thus, a MDDB can be viewed as a d-dimensional array, indexed by the values of the d dimension attributes, whose cells contain the values of the measure attributes for the corresponding combination of dimension attributes.

We consider a class of queries over data cubes, which we shall call range queries [3][5][8][9], that apply a given aggregation operation over selected cells where the selection is specified as contiguous ranges in the domains of some of the attributes. In particular, we will consider one type of aggregations: MAX/MIN. The corresponding range queries are termed range-max/min queries. For example, a range-max/min query of a data cube $\mathcal{DC}$ with d *dimension attributes* $a_1,\ldots,a_d$ and one *measure attribute* M can be described as follows:

SELECT MAX($\mathcal{DC}.M$) FROM $\mathcal{DC}$ WHERE
$l_1 \le \mathcal{DC}.a_1 \le h_1$ AND $l_2 \le \mathcal{DC}.a_2 \le h_2$ AND … AND $l_d \le \mathcal{DC}.a_d \le h_d$

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 467–476, 2000.

The most direct approach to evaluate a range-max/min query is to use the data cube itself, but the disadvantage of this approach is that the number of cells that need to be accessed is proportional to the size of the sub-cube defined by the query. So we need to find some other way to speed up rang-max/min queries. The main idea is to pre-compute multidimensional prefix-maxs of the data cube. Note that the direct approach can be applied to other aggregate functions such as SUM.

To handle the range query for the aggregate function SUM efficiently, considerable research has been done in the database community. In [8] and [5], a prefix sum approach and a relative prefix sum approach were presented respectively. The Prefix Sum approach use Prefix Cube ($\mathcal{PC}$), of the same size of $\mathcal{DC}$, as auxiliary information to answer range-sum queries. In particular, $\mathcal{PC}[x_1,\ldots,x_d]$ stores the sum of all the data in $\mathcal{DC}$ ranging from [0,…,0] to $[x_1,\ldots,x_d]$. With the use of $\mathcal{PC}$, any range-sum query of $\mathcal{DC}$ can be computed with a constant (2^d) cell accesses. Although the Prefix sum approach is very powerful. However the update cost is high. So an improved approach that has been recently proposed to balance the query-update tradeoff between the data cube and prefix cube is the relative prefix cube ($\mathcal{RPC}$) [5] which partitions the single large prefix cube into smaller ones.

But little work has been done with the range-max/min queries besides the one which is based on pre-computed max/min over a balanced hierarchical tree-based algorithm [8]. In this method, a generalized quad-tree is constructed on the data cube and in each tree node the index of the maximum value in the region covered by that node was stored. Then a branch-and-bound[10]-like procedure was used to speed up the queries. Due to the space constraints, the detail information could be found in [8].

Although the above-mentioned method, to some extent, can speed the process of finding the max of a range, it still has some drawbacks. Because of the larger size of the tree, it is very hard to load all the nodes of the tree into the main memory at the same time. So the speed of answering the range-max queries will be reduced by the frequent access of hard disk. In addition, the tree structure is not easily maintained especially when the dimensionality of the data cube and the size of each dimension are high. So here we present an efficient block-based algorithm for range-max/min. In this paper, we use the idea of relative prefix cube method to evaluate the range-max/min queries in data cubes, but of little difference. This approach achieves better performance than tree-based algorithm and also constrains update costs.

2 The Block-Based Algorithm for Range-max Queries

In order to reduce the chance of accessing hard disk and speed up the range-max queries, we present a new block-based algorithm. In our method, we also employ the same data cube model which is used in [5]. The original data cube is denoted as $\mathcal{DC}$. Two kinds of auxiliary information are used in our method: (1) One is called $\mathcal{BPC}$ (Block Pre-computed max Cube). (2) The other one is called $\mathcal{LPC}$ (Location Pre-computed Cube).

2.1 Block Pre-Computed Max Cube ($\mathcal{BPC}$)

Definition 1. A Data Cube $\mathcal{DC}$ of d dimension, is a d-dimensional array. For each dimension, the index can be ranged from 0 to $n_i - 1$ (n_i donates the size of the i^{th} dimension). So a cell in the data cube can be expressed in the following form:

$$\mathcal{DC}[x_1,\ldots,x_d] \quad \text{where} \quad 0 \le x_i \le n_i - 1$$

Example 1. Fig. 1 shows a 2 dimension data cube with each dimension size equal to 8.

Index	0	1	2	3	4	5	6	7
0	33	49	52	65	82	33	59	49
1	95	33	81	83	88	7	15	72
2	58	95	54	11	29	88	79	74
3	16	44	53	52	21	25	2	89
4	91	82	39	39	80	78	59	77
5	12	92	13	26	25	29	84	41
6	50	85	76	77	71	50	89	57
7	51	24	98	51	22	90	74	83

Fig. 1. A Data Cube $\mathcal{DC}$

Definition 2. Given a Data Cube $\mathcal{DC}$ of d dimension, and d integers $b_1,\ldots,b_d$ called partition factors, the **Block Pre-computed max Cube** of $\mathcal{DC}$, denoted as $\mathcal{BPC}$, is another data cube such that

1. It is a d-dimensional array with the same dimension sizes as $\mathcal{DC}$
2. It is partitioned into a number of sub-cubes called blocks by applying partition factor b_i on i^{th} dimension. The first cell, lowest cell index in each dimension, is called **anchor cell**. The last cell, highest cell index in each dimension, is called **end cell**.
3. Each cell $\mathcal{BPC}[x_1,\ldots,x_d]$ in $\mathcal{BPC}$ stores the maximum of all the cells $\mathcal{DC}[j_1,\ldots,j_d]$ where $\lfloor x_i / b_i \rfloor b_i \le j_i \le x_i$

Example 2. Fig. 2 shows a $\mathcal{BPC}$ of the given Data Cube $\mathcal{DC}$ in Fig. 1. The partition factors are set to 3. Note that $\mathcal{BPC}[4,4]$ holds the maximum value $\mathcal{DC}[i,j]$ where $3 \le i,j \le 4$. Also the maximum value is 80 which appears in the location (4,4) in $\mathcal{DC}$. Hence it is recorded in $\mathcal{BPC}[4,4]$.

Index	0	1	2	3	4	5	6	7
0	33	49	52	65	82	82	59	59
1	95	95	95	83	88	88	59	72
2	95	95	95	83	88	88	79	79
3	16	44	53	52	52	52	2	89
4	91	91	91	52	80	80	59	89
5	91	92	92	52	80	80	84	89
6	50	85	85	77	77	77	89	89
7	51	85	98	77	77	90	89	89

Fig. 2. $\mathcal{BPC}$ of $\mathcal{DC}$ in Fig. 1. Drawn for reference

2.2 Location Pre-computed Cube ($\mathcal{LPC}$)

Definition 3. Given a Data Cube $\mathcal{DC}$ of d dimension, and d partition factors $b_1,\ldots,b_d$, the **Location Priority** of a cell with index $x_1,\ldots,x_d$ in $\mathcal{DC}$, denoted as $\mathcal{LP}(x_1,\ldots,x_d)$, is an integer which has the value of

$$\prod_{i=1}^{n}(x_i - \lfloor x_i/b_i \rfloor b_i + 1)$$

Definition 4. Given a Data Cube $\mathcal{DC}$ of d dimension, and d partition factors $b_1,\ldots,b_d$, the **Location Pre-computed Cube** of $\mathcal{DC}$, denoted as $\mathcal{LPC}$, is another data cube such that

1. It has the same dimension d and
2. If the size of the i^{th} dimension in $\mathcal{DC}$ is n_i, i.e, it ranges from 0 to n_i-1, then the dimension i in $\mathcal{LPC}$ will be ranged from 0 to $\lceil n_i/b_i \rceil - 1$.
3. Each cell $\mathcal{LPC}[x_1,\ldots,x_d]$ in $\mathcal{LPC}$ stores two items
   - The max of all the cells $\mathcal{DC}[j_1,\ldots,j_d]$ where $b_i x_i \le j_i < \max(b_i(x_i+1),n_i)$ and
   - The location of one of the cells that hold this maximum value and the cell selected should have the maximum location priority among these cells that hold this maximum value.

We will denote the maximum value stored in $\mathcal{LPC}[x_1,\ldots,x_d]$ simply as $\mathcal{LPC}[x_1,\ldots,x_d]$.Max, and the maximum location as $\mathcal{LPC}[x_1,\ldots,x_d]$.MaxL.

Note that assume there are m cells which have the same value as the max value of a block, we will choose the cell's index to be stored in $\mathcal{LPC}$ which have the maximum location priority among these m cells. This is for efficient processing purpose. The detailed reason will be presented in the following section.

Example 3. Fig. 3 shows the $\mathcal{LPC}$ of the Data Cube $\mathcal{DC}$ in Fig. 1. The partition factors are set to 3. Note that $\mathcal{LPC}[0,1]$ holds the maximum value and its location among $\mathcal{DC}[i,j]$ where $0 \le i \le 2$and $3 \le j \le 5$. The maximum value is 88, and it appears in cell(1,4) and cell(2,5). $\mathcal{LP}(1,4)=(1-0+1)(4-3+1)=4$ is less than $\mathcal{LP}(2,5)=(2-0+1)(5-3+1)=9$. Hence Max 88 and MaxL (2,5) are stored in $\mathcal{LPC}[0,1]$.

Index	0	1	2
0	95|(2,1)	88|(2,5)	79|(2,6)
1	92|(5,1)	80|(4,4)	89|(3,7)
2	98|(7,2)	90|(7,5)	89|(6,6)

Fig. 3. $\mathcal{LPC}$ of Data Cube $\mathcal{DC}$ in Fig. 1

In many practical applications, because of the large size of $\mathcal{BPC}$, it is very hard to load all the cells of $\mathcal{BPC}$ into main memory during the process of answering range-max queries. But for $\mathcal{LPC}$, it is possible that it can be maintained in main memory even when $\mathcal{DC}$ is very large. A cell of $\mathcal{LPC}$ covers an area of the data cube $\mathcal{DC}$ equal to about b^d cells (without loss of generality, we assume partition factors $b_1=\ldots=b_d=b$). The storage required by $\mathcal{LPC}$ is thus significantly smaller than that used by the $\mathcal{BPC}$ and its number of cells is only about $1/b^d$ of the $\mathcal{BPC}$. Besides, space savings grow

larger as the partition factor b grows. For example, consider a 3-dimensional data cube $\mathcal{DC}$ which has a size of 10GB and the partition factor b is 10. The $\mathcal{LPC}$ only needs about 3*10GB/1000=30MB storage. Given suitable partition factors, it may be feasible to keep all the cells of $\mathcal{LPC}$ in main memory when answering ad hoc range-max queries at run time. So it will reduce the chance of accessing disk. In order to use less disk storage to store auxiliary information and at the same time to achieve comparatively good query performance result, we proposed another method, hierarchical compact cube in [9]. The average query time using hierarchical compact cube is bounded by a constant independent on the number of data in the data cube.

2.3 Block-Based Algorithm

We now present the block-based algorithm for computing the range-max based on the Block Pre-computed max Cube $\mathcal{BPC}$, the Location Pre-computed Cube $\mathcal{LPC}$ and the original Data Cube $\mathcal{DC}$, given the dimensionality of the data cube d, the partition factors $b_1,...,b_d$ for each dimension and the range-max query MAX($l_1:h_1,l_2:h_2,...,l_d:h_d$) : $l_i,h_i(1\le i\le d)$ stand for the lowest and highest bound of the range-max query in each dimension of the data cube respectively.

To compute MAX($l_1:h_1,l_2:h_2,...,l_d:h_d$), we will decompose its region into $\prod_{i=1}^{d}(\lfloor h_i/b_i\rfloor-\lfloor l_i/b_i\rfloor+1)$ disjoint sub-regions (sub-blocks) as follows. Intuitively, we partition the range in each dimension into $\lfloor h_i/b_i\rfloor-\lfloor l_i/b_i\rfloor+1$ adjoining sub-ranges where the middle sub-ranges are properly aligned with the block structure.

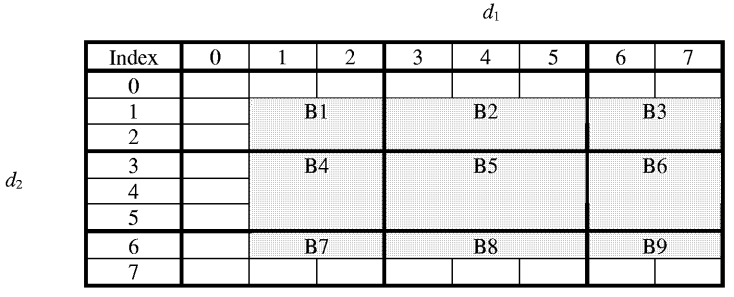


Fig. 4. Example of Range-max Query Region in $\mathcal{DC}$ in Fig. 1

For example Fig. 4 gives a pictorial example of the range-max query region (1:6,1:7)(represented by the shaded area). For dimension d_1, we get the following sub-ranges (1:2, 3:5, 6:6) and for dimension d_2, we get the following sub-ranges (1:2, 3:5, 6:7).

Then we can get all sub-blocks from the given range ($l_1:h_1,l_2:h_2,...,l_d:h_d$) by getting the combinations of the sub-ranges from each dimension, i.e. we select one sub-range from each dimension to form one block. All the combinations form all the

sub-blocks. For example in Fig. 4, We can get 9 combinations in all, i.e. 9 sub-blocks of the range region as shown in Fig. 4 (b). We then classify these sub-blocks into three types: **Whole Block** (WB) sub-blocks, **Start Non-Whole Block** (SNWB) sub-blocks and **Non-Start Non-Whole Block** (NSNWB) sub-blocks. The definitions for these three types of sub-blocks are listed as following:

Definition 5. Given a Data Cube $\mathcal{DC}$ of d dimension with each dimension sized of n_i $(1 \le i \le d)$, d partition factors $b_1,\ldots,b_d$ and a d-dimensional sub-block covered by a given range-max query, assuming the index of the anchor cell of the sub-block is $a_1,\ldots,a_d$ and the index of the end cell of the sub-block is $e_1,\ldots,e_d$.

1. It is a WB sub-block when it satisfies the following conditions:
   For each i $(1 \le i \le d)$
   Case (1) $\lfloor a_i/b_i \rfloor = \lfloor (n_i-1)/b_i \rfloor$: $e_i = n_i - 1$ and $a_i = \lfloor a_i/b_i \rfloor b_i$, or
   Case (2) $\lfloor a_i/b_i \rfloor \neq \lfloor (n_i-1)/b_i \rfloor$: $e_i - a_i + 1 = b_i$
2. It is a SNWB sub-block when it is not a WB sub-block and a_i mod $b_i = 0$ $(1 \le i \le d)$.
3. Otherwise, it is a NSNWB sub-block.

Example 4. In Fig. 4, B5, B6 are WB sub-blocks among all the sub-blocks which are covered by the given range-max query (1:6,1:7). B8, B9 are SNWB sub-blocks. B1, B2, B3, B4, B7 are NSNWB sub-blocks.

BLOCK-BASED RANGE-MAX QUERY ALGORITHM
Input: Query Region (l_1:h_1,…, l_d:h_d)
Output: MAX(l_1:h_1,…, l_d:h_d)
begin
step (1) c_max:= the smallest value of the measure attribute
step (2) //Process all sub-blocks ∈ WB
for each sub-block i in WB set **do**
if ($\mathcal{LPC}$[$W(i)$].Max> c_max) **then** c_max:=$\mathcal{LPC}$[$W(i)$].Max;
//$W(i)$ stands for the block which covers the sub-block i
step (3) //Process all sub-blocks ∈ SNWB
for each sub-block i in SNWB set **do**
if $\mathcal{LPC}$[$W(i)$].Max ≤ c_max **then** process next sub-block
else if $\mathcal{LPC}$[$W(i)$].MaxL is inside Query Region **then** c_max:=$\mathcal{LPC}$[$W(i)$].Max;
else // $\mathcal{LPC}$[$W(i)$].MaxL is outside Query Region
put sub-block i in a sorted list SL for processing later;
// Elements in SL are sorted in a descending order according to the $\mathcal{LPC}$[$W(i)$].Max
step (4) // Process all sub-blocks ∈ NSNWB
for each sub-block i in NSNWB set **do**
if $\mathcal{LPC}$[$W(i)$].Max ≤ c_max **then** process next sub-block
else if $\mathcal{LPC}$[$W(i)$].MaxL is inside Query Region **then** c_max:= $\mathcal{LPC}$[$W(i)$].Max;
else // $\mathcal{LPC}$[$W(i)$].MaxL is outside Query Region
put sub-block i in a sorted list NSL for processing later;
//Elements in NSL are sorted in the same way as SL

```
 step (5) // Process unprocessed SNWB list SL
  for each sub-block i in SL do
   if LPC[W(i)].Max<=c_max then goto step (6)
    else if BPC[end cell of i] > c_max then c_max:= BPC[end cell of i];
 step (6) // Process unprocessed NSNWB list NSL
  for each sub-block i in NSL do
   if LPC[W(i)].Max<=c_max then goto step (7)
   else
     begin
       access all cells of sub-block i in DC to find their max value  TmpMax ;
       if TmpMax > current _max then c_max:=TmpMax;
     end
 step (7) return c_max;
end;
```

Explanation of Block-Based Algorithm:
Different methods were adopted for processing different kinds of sub-block. For sub-blocks which belong to WB set, we can get the max of the sub-block directly from the *LPC* without accessing *BPC* or *DC*.

For sub-blocks which belong to SNWB set , we can check *LPC* first to see whether the MaxL of the block which covers the sub-block is inside the query region. If so, we can get the sub-block's max directly from *LPC* without accessing *BPC*. If not, we put this sub-block in a sorted list called SL to be processed later. SL is sorted in a descending order according to the Max of the block which covers the sub-block. Through sorting, we can reduce the cell accesses in *BPC*. For example, if there are two elements in SL: *a*([2,3], 6) and *b*([6,7], 4) and the value of c_max is 10. Obviously, c_max=10 is greater than 6 (the max value of block *a* which is the first element in the list SL), so we needn't to process the rest sub-blocks in the list. Through this way, we can reduce the chance of accessing *BPC* because as noted earlier in section 2.2, *LPC* is small enough and can be maintained in main memory. The speed of accessing a cell in *LPC* is faster than that of accessing a cell in *BPC* which is stored on disk.

For sub-blocks which belong to NSNWB set, firstly we do the same processing as what is done to the SNWB sub-blocks because through this way, we can also reduce the probability of accessing the data in Data Cube *DC*. As shown in the algorithm, the dominated query cost is cost in step (6) because sometime we need to access all the cells within the sub-block to get the sub-block's max in this step. How to reduce the number of cells accessed in this step is very important. So we wish more NSNWB sub-blocks cover the cell whose value and location is stored in *LPC*. Then we can get the max of the NSNWB sub-blocks directly from *LPC* by only one cell access without accessing more cells in *DC*. To reach this goal, when there are two or more cells in a block which have the same maximum value of the block, we will choose the one which satisfies the condition as illustrated in section 2.2. Because we find that when the location of the max value of the block is closer to the end cell of the block, the probability of the NSNWB sub-blocks' containing the max cell of the block is higher. When we construct the *LPC*, we don't need to care about SNWB sub-blocks. Because for this kind of sub-blocks we get the max with only up to one cell access cost.

Example 5. We now present a concrete example using the block-based algorithm. Shaded area in Fig. 4 shows the range-max query (1:6,1:7). Firstly, we process the WB sub-blocks: B5, B6 and we get c_max=89. Secondly we process the SNWB sub-blocks: B8, B9. For sub-block B8, $\mathcal{LPC}[W(B8)]$.Max=90 is greater than c_max=89 and $\mathcal{LPC}[W(B8)]$.MaxL is not inside the query region. Hence we put B8 in list SL, then we get SL=(B8). For sub-block B9, $\mathcal{LPC}[W(B9)]$.Max=89 is equal to c_max. So we don't need to process it. Similarly then we process the NSNWB sub-blocks: B1, B2, B3, B4, B7. For sub-block B1, $\mathcal{LPC}[W(B1)]$.Max=95 is greater than c_max=89 and $\mathcal{LPC}[W(B1)]$.MaxL is inside the query region. Hence we change c_max to 95. For sub-block B2, because $\mathcal{LPC}[W(B2)]$.Max=88 is less than c_max=95. So we don't need to process it. B3 and B4 are the same as B2. For sub-block B7, $\mathcal{LPC}[W(B7)]$.Max=98 is greater than the c_max=95 and $\mathcal{LPC}[W(B7)]$.MaxL is not inside the query region. Hence we put B7 in list NSL. Next we need to process SL. Because $\mathcal{LPC}$[W(B8)].Max=90 is less than c_max=95, we needn't process it. Then we process NSL. Because $\mathcal{LPC}$[W(B7)].Max=98 is greater than c_max=95, we need to access all the cells in B7 and get the TmpMax=85. So we don't need change the value of c_max. Note that this is the only time we need to access the data in the original data cube. Finally we get the answer of the range-max query: 95. We find that some sub-blocks are pruned when answering the range-max query.

3 Updates

Updating a cell in the Data Cube $\mathcal{DC}$ may require updates to $\mathcal{BPC}$ and $\mathcal{LPC}$. Updates in $\mathcal{BPC}$ are constrained to a region which is in one block. $\mathcal{BPC}$ cells within the block boundary that are greater in index than updated cell are affected. There is also potential update in $\mathcal{LPC}$ if the value of the update cell in $\mathcal{DC}$ is greater than the max value stored in the corresponding cell of $\mathcal{LPC}$ or if the value of the updated cell in $\mathcal{DC}$ is equal to the max value stored in the corresponding cell of $\mathcal{LPC}$, but the Location Priority of the updated cell is greater than the cell stored in the corresponding cell of $\mathcal{LPC}$. Suppose the updated cell is $\mathcal{DC}[x_1,\ldots,x_d]$, the minimum update cost in $\mathcal{BPC}$ is 0 cell and maximum update cost is $\prod[(\lfloor x_i/b_i \rfloor + 1)b_i - x_i]$ cells. Also the minimum and maximum update cost in $\mathcal{LPC}$ are 0 cell and 1 cell respectively. From the above, we know that the main update cost is produced when we update $\mathcal{BPC}$ due to one cell update in $\mathcal{DC}$. But it is only constrained to one block whose size is about b^d cells (assume all the partition factors are the same and equal to b).

4 Experiments and Performance Analysis

We randomly generated a set of data cubes by varying the data sizes, partition factors and dimensions. Without loss of generality, we assume data cubes $\mathcal{DC}$ with equal size

for each dimension. Accordingly, we constructed $\mathcal{BPC}$ and $\mathcal{LPC}$ of these data cubes. We then generated about 100,000 queries of random range and measure the average accessed cells for each data cube. The experiments are run in Linux and from these experiments we conclude several observations.

1. Given a d-dimensional data cube with each dimension size of n, our proposed algorithm performs best when the partition factor $b \approx \sqrt{n}$ which is independent of the dimensionality. The best partition factors for a 2-dimensional data cube, a 3-dimensional data cube, a 4-dimensional data cube with each dimension size of n=100, as shown in Fig. 5, are about 10. The best partition factors for 3-dimensional data cubes with each dimension size of n=50, 150, 200, as show in Fig. 6, are 7,12, 14 respectively.
2. Given a d-dimensional data cube with each dimension size of n, the tree-based algorithm's performance decreases when the partition factor (called tree fan-out in [8]) increases. In Fig. 7, the best partition factor is 2.
3. Our experiments also show that our proposed method outperforms tree-based algorithm as illustrated in Fig. 8 given the same Data Cube with the optimal partition factors chosen for each method. We can observe that with the cells covered by the range-max query increasing, the average accessed cells of our proposed method are increased very slowly.

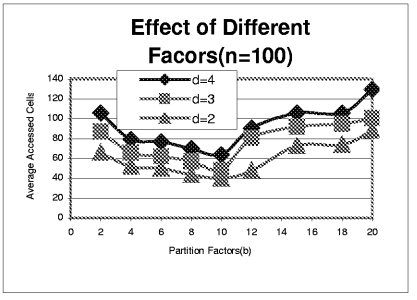


Fig. 5.

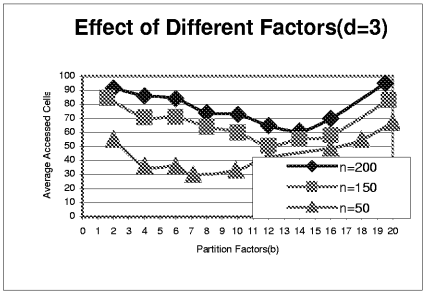


Fig. 6.

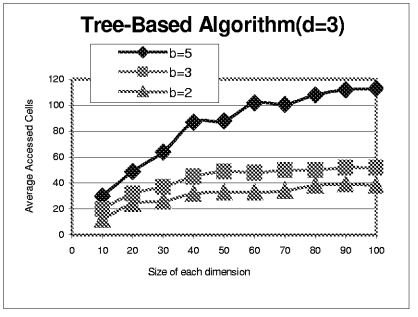


Fig. 7.

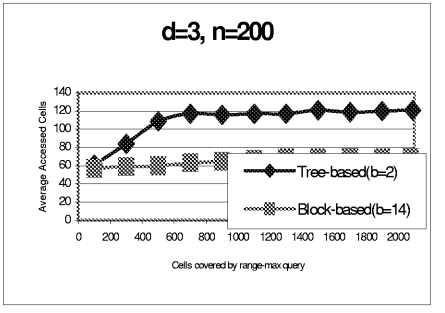


Fig. 8.

5 Conclusion

In this paper, we have presented an efficient algorithm for computing range-max queries on a data cube in an OLAP system. The essential idea for speeding up range-max queries is to use some pre-computed auxiliary information based on the data cube. In our proposed method, we use two kinds of auxiliary information which are called Block Pre-computed max Cube $\mathcal{BPC}$ and Location Pre-computed Cube $\mathcal{LPC}$ respectively. Also $\mathcal{LPC}$ is small enough to be loaded in main memory during answering ad hoc range-max queries and therefore we avoid frequent disk accesses and speed the range-max queries. The experiments showed our proposed method outperforms tree-based method.

References

1. R. Agrawal, A. Gupta. S. Sarawagi. Modeling multidimensional databases. In Proc. of the 13th Int'l Conference on Data Engineering, Birmingham, U.K., April 1997.
2. J. L. Bentley. Multidimensional divide and conquer. Comm. ACM, 23(4): 214-229,1980.
3. C. Y. Chan, Yannis E. Ioannidis. Hierarchical Cubes for Range-Sum Queries. Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, pp 675-686.
4. E. F. Codd. Providing OLAP(on-line analytical processing) to user-analysts: an IT mandate. Technical report, E.F. Codd and Associates, 1993.
5. S. Geffner, D. Agrawal, A. EI Abbadi, T. Smith. Algorithms for the Relative Prefix Sum Approach to Range sum Queries in Data Cubes. University of California, Santa Barbara, Computer Science Technical Report TRCS99-01.
6. J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Data cube: A relational aggregation operator generating group-by, cross-tabs and sub-totals. In Proc. Of the 12th Int'l Conference on Data Engineering, pp 152-159, 1996.
7. V. Harinarayan, A. Rajaraman, J. D. Ullman. Implementing Data Cubes Effieciently. Proceedings of ACM SIGMOD 1996 International Conference on Management of Dat, Montreal, Canda, June 1996 pp205-216.
8. C. Ho, R. Agrawal, N. Megiddo, R. Srikant. Range Queries in OLAP Data Cubes. In Proc. of the ACM SIGMOD Conference on the Management of Data, pp 73-88,1997.
9. S. Y. Lee, T. W. Ling, H. G. Li. Hierarchical Compact Cube for Range-Max. Proceedings of the 26th VLDB Conference Cairo, Egypt, September 2000.
10. L. Mitten. Branch and bound methods: General formulation and properties. Operations Research, 18:24-34, 1970.
11. The OLAP Concil. MD-API the OLAP Application Program Interface Version 0.5 Specification, September 1996.

Temporal Databases with Null Values

Mansik Park and Hans-Hermann Leinen

Department of Computer Science III, University of Bonn
Roemerstr. 164, 53117 Bonn, Germany
{park,leinen}@cs.uni-bonn.de

Abstract. In this paper we present a temporal data model capable of representing null values in valid time based on first normal form (1NF) and temporal intervals. As opposed to previous temporal data models, we store the length of valid time intervals in our model as additional information. In our model we can find temporal information efficiently with special temporal operators. In the case where temporal information is partially known, we can derive this information as complete temporal information or use extended algebraic operations to evaluate a query. Our model is implemented using C++.

1 Introduction

Research on incomplete information for the conventional relational model has been carried out extensively and many kinds of mechanisms have been proposed [Cod 79, Cod 86, Bis 81, Ges 90, IL 84, Lip 79, Vas 80, Zan 84]. In most models, missing information is treated with a null value which is given different semantics, such as value existent but unknown, value inapplicable, no information, etc.

Temporal databases store and handle not only the single (current) state of the real world but also past and future states of data. Most of the temporal data models represent temporal information through points, intervals, or sets of intervals [MS 91, TCG+ 93, Sno 95, Ste 98]. In practice, this temporal information often happens to be incomplete. Especially at the moment of data insertion there could be large amounts of only partially known temporal information. For example, we know only the beginning or the end point of some information, but not both. In the worst case, we know the data value but have no information when it could have occurred. When a temporal data model allows an unknown valid time, then users can insert all partially known information. To deal with this extended set of information, a proper mechanism must be provided.

It is often the case with temporal information that only the length of valid time is known. For example, do you know *exactly when* your last vacation began? Do you know *how long* it was? If we know the length together with one end point of an interval, we can derive complete temporal information from this. In making use of this idea, our model, unlike previous temporal data models, represents temporal information through intervals with length. Some temporal

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 477–487, 2000.

data models represent also intervals with duration (length). This duration is always linked to start- or end points in intervals. However, we present length as an explicit value in valid time.

This paper is organized as follows. In Section 2, we introduce an example that describes the motivation of our work. This example will be used throughout the paper. Section 3 presents our model. In Section 4, we define the algebra. Section 5 gives query examples and their evaluation, which show the power of our model. In Section 6, we give an overview of how our model is implemented. The last section provides a summary of this paper. Finally, we discuss some problems and future research.

2 Motivating Examples and Related Work

In this section, we present examples which demonstrate the problems of missing information in valid time and we mention related works which have also handled these problems. We have an employee table (E) as follows:

Name	Dep.	Salary	VT	L
Smith	A	3000	[1-5)	4
Smith	B	3500	[?-10)	?
Smith	A	4000	[11-?)	5
Tom	B	4000	[?-?)	?
Tom	B	4000	[?-?)	12

This example table shows that it could be useful to store the length of temporal intervals. In the third tuple we can derive the end point of the interval using length. At first glance, the two tuples for Tom appear like similar information, because without length they represent the same information. However, when a system is asked to 'List all employees having worked in department B definitely more than 10 (months, years)', we can find the results very efficiently through the value length. The other motivation for our model is, when data in a non-temporal database should be migrated to a temporal database, the use of null values (?) in valid time would be very useful. At first we can replace all temporal data with null values in valid time and update it step by step.

There are a few temporal data models concerned with temporal indeterminacy. In [CC 87] null values are introduced in valid time, but their use is limited and primitive. The model of [GNP 92] deals with temporal incompleteness and extends a temporal algebra that uses three-valued logic. However, this model, like [CC 87], is based on non-1NF and represents no temporal length. Furthermore, the user must always give a lower/upper-limit of temporal intervals. [DS 93] proposed representing the indefinite point of time through probability. This model provides no length of temporal information. The temporal data model in [CCP 97] describes an interval by start, end and duration where indeterminacy

is due to different granularities. This model does not allow the storage of unconstrained null values. A similar way of bounding null values is found in [BCTP 99, Kou 94]. In [BCTP 99], unbounded null values can be simulated by specifying '*lasting at least 0*', though. Null values in our model have no constraints and some extended relational operators make explicit use of length.

Research on temporal indeterminacy has also been performed by the artificial intelligence (AI) area. The AI area is more interested in deriving new knowledge from the given temporal information. The model of [MP 92] defined different modal operators to query temporal information. Recently, [ChC 97] dealt with temporal indeterminacy within the framework of deductive databases using Event Calculus [KS 86].

3 Modeling Valid Time with Null Values

3.1 Temporal Intervals with Length

There are three obvious parameters one can find for intervals: the beginning, the end and the length. Whereas beginning and end are restricted only by their domain (and the fact that a beginning should never be greater than its corresponding end), a stronger condition holds for the length, since an interval is expected to have non-negative length. In addition, the constraint length = end - beginning must hold for every interval. The following gives a formal description for this:

Let T be a set (the domain of time), G := (T,+) a group, $L \subseteq T$. Furthermore, let π_i be the i-th projection. $\mathcal{T} \subseteq T \times T \times L$ shall be the set of **temporal intervals** such that $\forall t \in \mathcal{T} : (\pi_2(t) - \pi_1(t)) \in L$.
We define the following functions:

$$\begin{aligned} \vdash: \quad \mathcal{T} &\to T \text{ (**beginning** of a temporal interval)} \\ (b,e,l) &\mapsto b \end{aligned}$$

$$\begin{aligned} \dashv: \quad \mathcal{T} &\to T \text{ (**end** of a temporal interval)} \\ (b,e,l) &\mapsto e \end{aligned}$$

$$\begin{aligned} |\cdot|: \quad \mathcal{T} &\to L \text{ (**length** of a temporal interval)} \\ (b,e,l) &\mapsto l \end{aligned}$$

For example, if t:=(1,4,3) is a temporal interval, then $\vdash(t) = 1$ is its beginning, $\dashv(t) = 4$ is its end, and $|\cdot|(t) = 3$ is its length. To prevent having ill-formed temporal intervals in our set, the specified length not being the distance between the beginning and the end of the interval, we restrict $\mathcal{T}$ to being a subset of $T \times T \times L$ such that $\forall t \in \mathcal{T} : \pi_3(t) = \pi_2(t) - \pi_1(t)$.
We will now extend the sets T and L to the value 'null'. The question-mark (?) will be used to denote this value.

To extend our sets T and L to 'null', we introduce $T_? := T \cup \{?\}, L_? := L \cup \{?\}$ and define:

$$+ : T_? \times T_? \to T_?,$$

$$+(\mathrm{a,b}) = \begin{cases} a+b & \text{if } a,b \in T, \\ ? & \text{else} \end{cases}$$

It is now possible to extend the set of temporal intervals, $\mathcal{T}$, to our 'null' value: $\mathcal{T}_? \subseteq T_? \times T_? \times L_?$. To make the three operations given above work correctly, we must extend them, too:

$$\vdash : \mathcal{T}_? \to T_?, \qquad \vdash (b,e,l) = \begin{cases} b & \text{if } b \in T, \\ (e-l) & \text{if } e \in T, l \in L, b \notin T, \\ ? & \text{else.} \end{cases}$$

$$\dashv : \mathcal{T}_? \to T_?, \qquad \dashv (b,e,l) = \begin{cases} e & \text{if } e \in T, \\ (b+l) & \text{if } b \in T, l \in L, e \notin T, \\ ? & \text{else.} \end{cases}$$

$$|\cdot| : \mathcal{T}_? \to L_?, \qquad |\cdot|(b,e,l) = \begin{cases} l & \text{if } l \in L, \\ (e-b) & \text{if } e,b \in T, l \notin L, \\ ? & \text{else.} \end{cases}$$

3.2 Classification of Null Values

To basically classify temporal intervals regarding their null values, we remark that a temporal interval may consist of either zero, one, two or three null values. This leads to the following definition:

$$\begin{aligned} \#_? : \mathcal{T}_? &\to \mathbb{N} \\ t &\mapsto |\{i \in \mathbb{N} : \pi_i(t) = ?\}|, \ (1 \le i \le 3). \end{aligned}$$

The function given above simply counts the number of ?'s in a temporal interval. For example, $\#_?(1,?,?) = 2$. We can now define

$$\mathcal{T}_?^p := \{t \in \mathcal{T}_? : \#_?(t) = p\}$$

as the subset of $\mathcal{T}_?$ having p null values. Note that $\mathcal{T} = \mathcal{T}_?^0 \dot\cup \mathcal{T}_?^1 \dot\cup \mathcal{T}_?^2 \dot\cup \mathcal{T}_?^3$. We can omit $\mathcal{T}_?^0$ by arguing that no user will specify all three parameters of a temporal interval, since each interval is already uniquely specified by two parameters. Consider the following function:

$$\begin{aligned} &\gamma : \mathcal{T}_?^1 \to \mathcal{T}_?^0 \\ &t \mapsto (\vdash(t), \dashv(t), |\cdot|(t)) \end{aligned}$$

By applying this function to each temporal interval after input, we can completely eliminate elements in $\mathcal{T}_?^1$. Nevertheless, the user can conveniently enter temporal intervals in a common way.

3.3 Evaluation of 'Null'

To be able to compare valid-time values, it is important to intergrate null into the equality relation. Since we do not have further information about null values, there are two alternatives:

1. A null value is equal to every other value in the domain.
2. A null value is different from any other value in its domain, including other null values.

The first method is often called 'maybe' semantics, because it *may be* that the null value will be replaced by a value against which it is being compared. In contrast, the second alternative is often called 'sure' semantics.

The following section will exploit both semantics. To distinguish between the 'maybe' and 'sure' versions of any operator, we will mark the corresponding operator with an 'm', respectively 's'.

3.4 Mapping Temporal Intervals to $\mathbb{R}$

Some operators of the relational algebra (see Section 4) require a corresponding representation of our intervals in $\mathbb{R}$. To give every temporal interval a proper representation as a set in $\mathbb{R}$ (even those containing null values), we introduce a special mapping:

$\iota^m : \mathcal{T}_? \to \mathcal{P}(\mathbb{R})$, where $\mathcal{P}(\mathbb{R})$ is the power set of $\mathbb{R}$.

$$\iota^m(t) = \begin{cases} \left[\vdash(t), \dashv(t)\right) & \text{if } \#_?(t) = 0 \\ \left[\vdash(\gamma(t)), \dashv(\gamma(t))\right) & \text{if } \#_?(t) = 1 \\ \iota_2^m(t) & \text{if } \#_?(t) = 2 \\ \mathbb{R} & \text{if } \#_?(t) = 3 \end{cases}$$

$\iota^s : \mathcal{T}_? \to \mathcal{P}(\mathbb{R})$

$$\iota^s(t) = \begin{cases} \left[\vdash(t), \dashv(t)\right) & \text{if } \#_?(t) = 0 \\ \left[\vdash(\gamma(t)), \dashv(\gamma(t))\right) & \text{if } \#_?(t) = 1 \\ \iota_2^s(t) & \text{if } \#_?(t) = 2 \\ \varnothing & \text{if } \#_?(t) = 3 \end{cases}$$

The functions ι_2^m and ι_2^s are given in the table below. For the definition, let $x \in T$, $X := \{x\}$, $y \in L$.

t	$\iota_2^m(t)$	$\iota_2^s(t)$
$(x, ?, ?)$	$[x, \infty)$	X
$(?, x, ?)$	$(-\infty, x)$	X
$(?, ?, y)$	$\mathbb{R}$	$\varnothing$

The important thing to note about ι is that the information on length is completely ignored, when the interval lies in $\mathcal{T}_?^2$. For the 'maybe' version, there is no other reasonable solution than to map to $\mathbb{R}$. Since we want to match the interval with every other, this mapping is plausible for the 'maybe' semantics we intended. For the 'sure' version, the result set is the empty set. Like the 'maybe' version, this ignores the length completely, but ensures we get no match where we would not want one.

4 Algebraic Operation

In this section we define an algebra which operates on our model. Our algebra consists of two parts. The first part contains basic operations such as selection, projection, join, etc. In the second part special operations are defined which operate especially on our model and extend the already developed operators such as Time-slice. We integrate the concept of two versions of results for the evaluation of a query [Cod 86, Gra 79, Lip 79], i.e. a maybe and a sure version. However, we use only two-valued logic for both of the two versions.

4.1 Basic operation

Selection: Selection is defined straightforwardly as a conventional relational model as follows: σ_P (R) := {t| t ∈ R ∧ P(t) = true}, where the predicate P denotes a selection condition. However, there are two types of selection conditions in temporal databases, i.e. data selection condition (P_d) and temporal selection condition (P_{vt}). The predicate of data selection has the form <A θ a>, where A is an attribute name, a is a value, and $\theta \in \{=, \neq, <, >, \leq, \geq\}$.

The predicate of temporal selection is defined as follows:
P_{vt} := < VT θ_1 I > | < VT θ_2 X>, where VT is valid time, I is an interval, I $\in \mathcal{T}_?^0$, θ_1 is any one of Allens 13 operators [All 83], X is a point in time, θ_2 is any one of the set comparison ($\in, \subset, \subseteq$, set-equality, etc.). Because we have null values(?) in valid time, we define two types of results to a query: maybe- (P_{vtm}) and sure version (P_{vts}). $P_{vt} \in \{P_{vtm}, P_{vts}\}$.
We present an example for an operator = in < VT θ_1 I >.

$< VT = I >$	Maybe(P_{vtm})	Sure(P_{vts})
$[?-?) = [a-b)$	T	F
$[a-?) = [a-b)$	T	F
$[?-b) = [a-b)$	T	F
$[a-b) = [a-b)$	T	T
otherwise	F	F

The other operators behave like the above operator. Now we can define a predicate P of selection as follows; P:= $P_d | P_{vt} | P_d \wedge P_{vt}$.

Projection: π_X (R):={t(X) | t ∈ R}, where R is a relation on scheme S, t is a tuple with scheme X and X is a subset of S (X ⊆ S).
Projection retains all the valid time values regardless of null values like standard projection. Coalescing of tuples with equal values is carried out only for valid times without null values.

Difference: $R_1 - R_2 := \{t \mid \exists t_1 \in R_1, t_2 \in R_2$: $\mathcal{A}(t_1) = \mathcal{A}(t_2) = \mathcal{A}(t) \wedge (\iota(t.VT) = \iota(t_1.VT)$ - $(\iota(t_1.VT) \cap \iota(t_2.VT)))\}$, where ι is defined in Section 3, and $\mathcal{A}$ is defined as follows: $\mathcal{A}$(t) := set of snapshot-attributes of t, where t is a tuple. We also define a maybe and a sure version for the difference operation:

Operator (-)	Maybe	Sure
$[?-?) - [?-?)$	$\emptyset$	$\emptyset$
$[? - a) - [? - b)$	$\emptyset$ if a≤b, $[b - a)$ if $a > b$	{a} if a≠b, $\emptyset$ else
$[? - a) - [b-?)$	$[? - Min(a, b)]$	{a} if a≠b, $\emptyset$ else
$[a-?) - [? - b)$	$[b-?)$ if $a < b$, $[a-?)$ if a≥b	{a} if a≠b, $\emptyset$ else
$[a-?) - [b-?)$	$[a - b)$ if $a < b$, $\emptyset$ if a≥b	{a} if a≠b, $\emptyset$ else

Union: $R_1 \cup R_2 := \{t \mid \exists t_1 \in R_1, t_2 \in R_2$: $\mathcal{A}(t_1) = \mathcal{A}(t_2) \wedge (\iota(t_1.VT) \cap \iota(t_2.VT) \neq \emptyset) \wedge (\iota(t.VT) = \iota(t_1.VT) \cap \iota(t_2.VT))\}$, where the same conditions as in the difference operation are available. A maybe-union ($\cup_m$) and a sure-union ($\cup_s$) are the same in principle as in the above difference operator.

Join: $R_1 \bowtie_\theta R_2 := \{t \mid t(R_1 \cup R_2) \wedge \theta(t_1, t_2)\}$, where t_1, t_2 are tuple in relation R_1, R_2 respectively. Without loss of generality we consider θ-join. We use an intersection semantic for a valid-time join. Therefore, after join operation there is an overlap in valid time of R_1 and R_2. We define maybe-join $\bowtie_\theta^m$ and sure-join $\bowtie_\theta^s$, in case a null value in valid time occurs. The results of $\bowtie$-evaluation for valid time in intersection semantic is as follows.

VT_1	VT_2	Maybe	Sure
$[?-?)$	$[?-?)$	$[?-?)$	$\emptyset$
$[?-?)$	$[? - a)$	$[? - a)$	$\emptyset$
$[?-?)$	$[a - b)$	$[a - b)$	$\emptyset$
$[a-?)$	$[b-?)$	$[a - b)$ if $a < b$, {a} if a=b, $[b - a)$ if $a > b$	{a} if a ≤ b, {b} if $a > b$

4.2 Special operation

We use the functions of Section 3 as operators for temporal information. For example, to find a beginning of a temporal interval from any tuple, we can use $\vdash$, and to find an end of a temporal interval from any interval, we use $\dashv$ for the type $\mathcal{T}_?$.

f(VT) θ **t**, where f ∈ {$\vdash$, $\dashv$, $|\cdot|$}, $\theta \in \{=, \neq, <, >, \leq, \geq\}$ and t is a point in time. For instance, we use $\vdash$(VT) < 10 or $\dashv$(VT) = 12 for a selection predicate.

We can find a particular length of an interval, for example length 5, directly with an expression $|\cdot|(\mathrm{VT}) = 5$. We can also combine all the operators with the other selection predicates. For instance, a predicate could be as follows:
(Name =Tom) $\bigwedge$ $(\dashv(\mathrm{VT}) < 20)$ $\bigvee$ $(\vdash(\mathrm{VT}) = 12)$ $\bigwedge$ $(|\cdot|(\mathrm{VT}) = 5)$.

In practice, there are many cases in which we want to find all the tuples of a relation whose valid time is either known or unknown. The function $\#_?$ from Section 3 can be used for that. We define two operators **VT-known(VT-κ), VT-unknown(VT-$\neg\kappa$)** as follows:

VT-κ(R) := {t ∈ R| $\#_?$(t) ∈ {0,1}}, where t is a temporal interval. The set of $\mathcal{T}_?^0$ and $\mathcal{T}_?^1$ belongs to VT-κ.

VT-$\neg\kappa$(R) := {t ∈ R| $\#_?$(t) ∈ {2,3}}, where t is a temporal interval. The set of $\mathcal{T}_?^2$ and $\mathcal{T}_?^3$ belongs to VT-$\neg\kappa$.

Move(μ): Sometimes we need to move a temporal interval, that is, given an existing interval we want an interval with the same length but in another time. For instance, several periodic events can be inserted or compared. To conveniently handle such tasks, we define a move-operation. Remember that we extended + to $\mathcal{T}_?$, so that we can add values in $\mathcal{T}_?$ and T.

$$\begin{aligned} \mu : \mathcal{T}_? \times T &\to \mathcal{T}_? \\ (t, n) &\mapsto (\vdash(t) + n, \dashv(t) + n, |\cdot|(t)) \end{aligned}$$

Time-slice(TS): When it is only necessary to find a (particular) limited time period, we use a time-slice operator. To carry out this operation on a temporal interval, we use the fuction ι from Section 3. Time-slice operation TS(R,v) is defined as follows:
TS(R,v):= {t | $\iota(t.VT) \subseteq \iota(v) \wedge (\exists$ s ∈ R :$\mathcal{A}$(t) = $\mathcal{A}$(s) $\wedge$ ($\iota(t.VT) \cap \iota(s.VT) \subseteq \iota$(v))) }, where R is a relation, v is an interval, t and s are tuples, ι is defined in Section 3. For sets of intervals, we define UTS(R,V):= $\bigcup_{v \in V}$TS(R,v).

Periodic Time-slice(PTS): It is practical to use an operator for a comparison over a particular time period: for example, the comparison of an employee's salary in this year with his salary five years ago. Therefore we extend the time-slice operation (TS) for that as follows.
PTS(R,v,n,k):= $\bigcup_{i=1}^{n}$ TS(R, μ(v,i*k)), where R is a relation, v is an interval(valid-time), k is a value for move(shift), n says how often the time-slice should be carried out.

When(Ω): It is useful to find timestamps in relations directly by an operator. The results of such an operation can also be used as a parameter for the other operation. The when operator is defined as follows:
Ω(R) := {v ∈ $\mathcal{T}_?$ | ∃ s ∈ R: s.VT = v}, where v is a temporal interval.

5 Query Examples and Evaluation

In this section we show the applicability of our algebraic operators to data queries by means of the following query examples. We assume that we have the same table (E) as in Section 2. The evaluation of query shows the power of our model.

Query 1: Find the salary of Tom at 8.
$\pi_{salary}(\sigma_{name=Tom \wedge VT \ni 8}(\mathrm{E}))$

Query 2: Give salary of Tom, when Smith worked in department A.
$\pi_{salary}(\sigma_{name=Tom}(\mathrm{UTS}(\mathrm{E}, \Omega(\sigma_{dep=A \wedge name=Smith}(E)))))$

Query 3: When did Tom work in the department that Smith worked in at the point of time 33 ?
$\Omega(\pi_{E_1}(\rho_{E_1}(\sigma_{name=Tom}(\mathrm{E})) \bowtie_{E_1.dep=E_2.dep} \rho_{E_2}(\sigma_{name=Smith \wedge VT \ni 33}(\mathrm{E}))))$
(ρ is a renaming of attributes.)

Query 4: When did Tom earn more than Smith ?
$\Omega((\rho_{E_1}(\sigma_{name=Smith}(\mathrm{E}))) \bowtie_{E_1.sal<E_2.sal} (\rho_{E_2}(\sigma_{name=Tom}(E))))$.

Query 5: Give all the data of which the valid time is unknown.
VT-$\neg\kappa$(E)

Query 6: Give the names of employees who have worked in any department more than 10 (months) or whose salaries did not change in more than 5 (months) ?
$\pi_{name}(\sigma_{|\cdot|(VT)>10}(\pi_{name,department}(\mathrm{E}))) \bigcup \pi_{name}\ (\sigma_{|\cdot|(VT)>5}(\pi_{name,salary}(E)))$

Query 7: Display and compare the salary of each employee in the year 1999 and 2000. (Assumed, valid-time granularity is a year.)
$\pi_{name,salary}(\mathrm{PTS}(\mathrm{E},[1999\text{-}2000),1,1))$

Query 8: List employees of the department A who began working before 5 and stopped working after 10.
$\pi_{name}(\sigma_{(\vdash(VT)<5) \wedge (\dashv(VT)>10) \wedge (dep=A)}(E))$

6 Implementation

A basic implementation of our model stores the length as an additional value in each valid time interval. Each time an interval is changed, consistency checks are applied after the change to ensure that the interval is still in a valid state. Finally, an implementation of γ eliminates any superfluous null value. This basic implementation suffers from massive memory and runtime overhead.
A more elaborate implementation utilizes the observation that we only need to store two parameters of each temporal interval. This reduces memory costs, but it should be noted that we cannot completely eliminate additional memory needs.

For example, let t =(1, ?, ?). Assuming that the implementation stores valid-time values in (begin, length) tuples (and therefore calculates the end when necessary), the internal representation of t must be altered when its value is changed to (?, 2, ?), leaving only t's end point in a known state.

To solve this problem, while not deviating from storing valid time as pairs, we observe that the only additional information we need to store is the type information of the valid time. Given a 'default' type, which in our example is the type (begin, length), we can mark all valid-time values that must be stored in an alternative type. This can for example be done in a separate 'type info' table, containing an index to the valid time being referred to and the type information itself. Note that the type information needs to be only one bit.

Since not every relation in a database needs to deal with the same degree of indeterminacy, the decision as to using the 'standard' valid-time intervals (using (begin,end) pairs) or the more complex intervals presented in this article could be left to the user.

Note that, in general, care has to be taken when synchronizing between redundant data such as the explicitly stored length and the implicit length given by begin/end-pairs. The basic implementation of our model in C++ can be requested.

7 Summary and Discussion

In this paper, we have presented a temporal data model capable of representing null values in a temporal interval. Because our model is based on 1NF and intervals, we can extend existing temporal relational models by our concept of modeling. By utilizing the length of valid-time intervals, we introduce useful features in temporal databases. Especially, allowing null values in valid time is advantageous for obtainig and presenting more temporal information.

Though we showed the power of our special temporal operator on a query evaluation, it is still unsatisfactory in obtaining results for two-version semantics. For instance, in join operation the result set of maybe semantic can become too big, on the other hand, the result set of sure semantic can become too small. In practice, the gap between those two extremes may be too wide. Therefore, it is desirable to get more specific versions of both maybe and sure semantics.

Null values in our model could be marked. This leads to distinct null values. Making those values comparable by applying temporal constraints gives the opportunity for temporal reasoning such as [BCTP 99]. It is also possible to mark null values in our model without any direct reference to the time axis. Especially by marking length information, we can compare those length values. In this way we could introduce more refined maybe- and sure versions.

We have omitted the case of incomplete data values in our model, because the treatment of this problem has already been sufficient and these results could be integrated in our model. However, it will be interesting to look for ways to deal with values of type $\mathcal{T}_?^3$ in future. Furthermore, an efficient storage technique for length should be developed.

References

[All 83] J.F.Allen: *Maintaining Knowledge about Temporal Intervals.* Communications of the ACM 16, 832-843, 1983.

[BCTP 99] V.Brusoni, L.Console, P.Terenziani, B.Pernici: *Qualitative and Quantitative Temporal Constraints and Relational Databases: Theory, Architecture, and Applications.* IEEE Trans. on Knowledge and Data Engineering, 11(6), 948-968, 1999.

[Bis 83] J.Biskup: *A foundation of Codds relational maybe-operations.* ACM Transactions on Database Systems 8, 608-636, 1983.

[CC 87] J.Clifford, A.Croker: *The Historical Relational Data Model(HRDM) and Algebra based on Lifespan.* In Proc. of the Int. Conf. on Data Eng. IEEE, 1987.

[ChC 97] L.Chittaro, C.Combi: *Temporal Indeterminacy in Deductive Databases;an approach based on event calculus.* (ARTDB-97), LNCS 1553, 1997.

[CCP 97] C.Combi, G.Cucchi, F.Pinciroli: *Applying Object-Oriented Technologies in Modeling and Querying temporally-oriented Clinical Database dealing with Temporal Granularity and Indeterminacy.* IEEE Transactions on Information Technology in Biomedicine, 1(2):100-127, 1997.

[Cod 79] E.F.Codd: *Extending the database relational model to capture more meaning.* ACM Transactions on Database Systems 4, 397-434, 1979.

[Cod 86] E.F.Codd: *Missing information in relational databases.* ACM SIGMOD 15, 53-77, 1986.

[DS 93] C.E.Dyreson, R.T.Snodgrass: *Valid-time Indeterminacy.* Proc. of the IEEE International Conference on Data Engineering, 335-343, 1993.

[Ges 90] G.H.Gessert: *Four valued logic for relational database systems.* ACM SIGMOD RECORD 19, 29-35, 1990.

[GNP 92] S.K.Gadia, S.S.Nair, Y.C.Poon: *Incomplete information in relational temporal databases.* Proc. of the 18th VLDB Conference, 395-406, 1992.

[Gra 79] J.Grant: *Partial values in a tabular database model.* Information Processing Letters 9, 97-99, 1979.

[IL 84] T.Imielinski, W.Lipski: *Incomplete information in relational databases.* Journal of the ACM 31, 761-791, 1984.

[Kou 94] M.Koubarakis: *Database Models for Infinite and Indefinite Temporal Information.* Information Systems, 19(2), 141-173, 1994.

[KS 86] R.Kowalski, M.Sergot: *A logic-based Calculus of Events.* New Generation Computing 4, 67-95, 1986.

[Lip 79] W.Lipski: *On Semantic Issues connected with Incomplete Information Databases.* ACM Transactions on Database Systems 4, 262-296, 1979.

[MP 92] R.Maiocchi, B.Pernici: *Automatic Deduction of Temporal Information.* ACM Transactions on Database Systems 17, 647-688, 1992.

[MS 91] L.Mckenzie, R.T.Snodgrass: *Evaluation of Relational Algebras incorporating the Time Dimension in Databases.* ACM Computing Surveys, 23, 501-543, 1991.

[Sno 95] R.T.Snodgrass editor: *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[Ste 98] A.Steiner: *A Generalisation Approach to Temporal Data Models and their Implementations.* PhD thesis, Swiss Federal Institut of Technology Zurich, 1998.

[TCG+ 93] A.Tansel, J.Clifford, S.Gadia, et al: *Temporal Databases; Theory, Design, and Implementation.* Benjamin/Cummings Publishing, 1993.

[Vas 79] Y.Vassiliou: *Null values in data base management; a denotational approach.* Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 162-169, 1979.

[Zan 84] C.Zaniolo: *Database relations with null values.* Journal of Computer System Science 28, 142-166, 1984.

Improving Temporal Joins Using Histograms

Inga Sitzmann and Peter J. Stuckey

Department of Computer Science and Software Engineering
The University of Melbourne
Parkville 3052, Australia
(inga, pjs)@cs.mu.oz.au

Abstract. Histograms are used in most commercial database systems to estimate query result sizes and evaluation plan costs. They can also be used to optimize join algorithms. In this paper, we consider how to use histograms to improve the join processing in temporal databases. We define histograms for temporal data and a temporal join algorithm that makes use of this histogram information. The join algorithm is a temporal partition-join with dynamic buffer allocation. Histogram information is used to determine partition boundaries that maximize overall buffer usage. We compare the performance of this join algorithm to temporal join evaluation strategies that do not use histograms, such as a partition-based algorithm based on sampling and a partition-join using the Time Index, an index structure for temporal data. The results demonstrate that the temporal partition-join is substantially improved through the incorporation of histogram information, showing significantly better performance than the sampling-based algorithm and achieving equivalent performance to the Time Index join without requiring an index.

1 Introduction

The increasing need to store time-related data has challenged conventional database technologies. Temporal information usually leads to a larger amount of data to be stored and processed. Query evaluation has to deal with different operators and different properties of temporal data. As in relational databases, efficient evaluation of the join operator is crucial for the overall performance of the system.

Several temporal join algorithms have been presented in the literature. Temporal joins are categorised in [GS91], taking the physical organisation and indexing of the data into account. A sort-merge-based algorithm for time-oriented data is presented in [PJ99]. An incremental approach is used to minimize the cost for updating temporal joins in append-only databases. In [RF93], several refinements of the temporal nested-loop join are proposed that minimize comparisons and buffer space required to reduce disk I/O. A partition-based temporal join is presented in [SSJ94]. Sampling is used to approximate the distribution of the data and to compute the partitioning. The *Time Index* is used in [SE96] for a temporal join. The information stored in the *Time Index* is used for a partition-based join. In the paper [LM92], a partition-based join in a multi-processor environment is described.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 488–498, 2000.

We suggest extending the use of histograms to temporal joins and present a partition-based temporal join using histogram information. Histograms have become standard in most commercial database systems for estimating result sizes and optimizing evaluation plans. Making use of information already available in the system allows us to improve temporal joins significantly at no additional cost. We have also investigated a histogram-based merge-join for append-only databases [SS00] which improved on a greedy temporal merge-join in some cases and showed more robustness overall.

2 Temporal Data and Joins

Time is assumed to be a sequence of discrete time instances which differ by exactly one minimal time unit (chronon). An entity in a temporal relation consists of a surrogate attribute S, time-varying attributes and time-invariant attributes. For each object in the database, different versions with different time intervals can exist. Surrogate attributes are used in temporal databases to relate different versions of an entity to the entity. Various temporal data models have been presented in the literature [SS88]. We use the time-interval representation where each tuple t is stamped with a time interval $[t.t_s, t.t_e]$ indicating the range of time where the tuple is valid.

The time attributes associated with temporal relations can refer to either the *valid time* or the *transaction time*. A classification of temporal joins is given in [GS91]. In this paper, we focus on evaluation of the valid-time time join. The *time* join finds tuples with intersecting time intervals without specifying a join predicate on non-time attributes. The *time-equi* join also requires equality on non-time attributes. Our approach can also be used for *time-equi* joins if joining on the time attribute first leads to more efficient evaluation [SS00].

3 Histograms

Histograms are statistical approximations to data distributions. They are used in database systems to estimate the size of a query and to find efficient access plans. Histograms are created by partitioning the distribution of attribute values into buckets. In each bucket, information about the estimated frequency of bucket values is maintained. To update histograms, samples of the relations are drawn.

Extensive research has been performed on creating and maintaining accurate histograms. Various classes of histograms have been presented. Two well-known classes of histograms are *equi-width* and *equi-depth* histograms. In an equi-width histogram, the widths of all buckets are the same. In equi-depth histograms, the number of tuples with the value represented by the bucket are the same. The width of the buckets varies. Examples for more advanced classes of histograms are *spline-based*, *V-optimal* and *end-biased* histograms. An extensive investigation of histogram classes can be found in [Poo97], [PIHS96].

3.1 Temporal Histograms

To estimate the number of tuples qualifying for a temporal join, we need to know the number of tuples active during a time interval. We distinguish between two kinds of tuples active in a time interval: (i) tuples starting in the interval and (ii) tuples that started in a previous interval but are still active in this interval. We call the latter "long-lived" tuples.

Temporal histograms show the number of tuples that start (resp. are still active) in each histogram bucket. In our experiments we use simple equi-width histograms. The use of more advanced histogram classes leads to higher accuracy for skewed data and should improve the performance of our evaluation methods even further.

Temporal histograms divide the time range of the relation P into m buckets where bucket i denotes time range $[H_P[i], H_P[i+1][$ [1] (assuming $H_P[m+1] =$ end_time is greater than all possible times). A temporal tuple t with start time $t.t_s$ and end time $t.t_e$ is active in histogram bucket i if one of the following conditions holds:

(1) $H_P[i] \leq t.t_s < H_P[i+1]$
(2) $t.t_s < H_P[i] \leq t.t_e$

The two histograms used in the algorithms represent:

1. the number of tuples $START_R[i]$ of R that start during bucket i (active due to the condition (1) above); and
2. the number of tuples $ACT_R[i]$ of R that started in a previous bucket j with $j < i$ but are still active in bucket i (active due to the condition (2) above).

Notice that the two histograms for the same relation have the same bucket boundaries.

Throughout, we assume that the histogram heights are normalized to the size of the actual relation. For example, if 2% of the tuples of relation P are sampled to create the histogram, the resulting start and active numbers are multiplied by 50 to give the histograms for the relation P.

Figures 1 (a) and (b) show histograms for the start times for two relations R and S. The histogram for R is equi-width, while that for S is equi-height. For example, 7 tuples in R start in the range [20,40[while 4 tuples in S start in the range [16,24[.

Information about "long-lived" tuples is summarized by giving the number of tuples still active from previous intervals at the start point of the histogram interval as shown in Figures 1 (c) and (d). For example, 4 tuples in R are still active at time 60 and possibly stay active till the end of the interval.[2]

[1] $[a, b[$ denotes the interval $\{x \mid a \leq x < b\}$

[2] Although they might have become inactive during the interval, they will be treated as if they are active during the whole duration of the interval (the worst case).

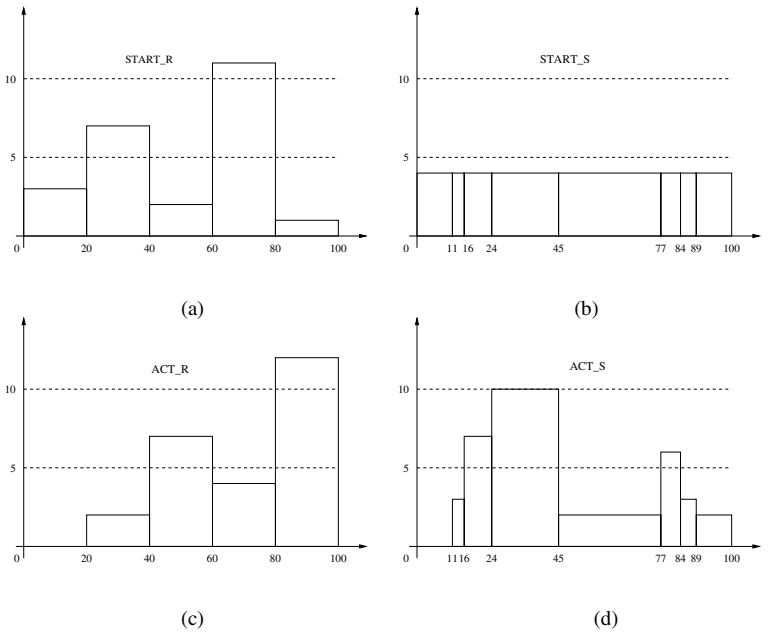


Fig. 1. Start times and active tuple histograms for an (a) equi-width histogram for relation R and (b) (start time) equi-height histogram for relation S

4 Temporal Partition-based Joins

Partition-based joins often have reduced I/O-costs in comparison with other join algorithms, e.g. the nested-loop join. Partition-based joins try to cluster tuples with similar values of the join attribute. The relation is then split into partitions that fit into main memory. During join evaluation, only tuples in corresponding partitions have to be compared.

A partition-based join consists of three phases. The first phase calculates the boundaries of the partitions. In the second phase, the relations are physically partitioned. This involves reading the relations once and writing the tuples back in the order of the partitions they belong to. The third phase finally reads the corresponding partitions of both relations and joins them in main memory.

The performance of a partition-based join strongly depends on the quality of the partitioning. A good partitioning avoids buffer overflow and requires the partitions to be read only once. Common techniques use sampling or an index to approximate the distribution of the attribute values used for calculating the partitioning. The disadvantage is that either additional block accesses for sampling are necessary or the data has to be stored in a particular index structure.

4.1 Partition-based Valid-time Join

The partition-based valid-time join (PBVTJ) was proposed in [SSJ94]. Its partitioning algorithm uses sampling to determine the partition boundaries. Their experiments showed that this join outperforms nested-loop and sort-merge-joins.

The buffer allocation scheme used in the approach of [SSJ94] divides the available buffer into an inner partition area of one block, a tuple cache page of one block for long-lived tuples of the inner relation, and an outer partition area (the remaining blocks). The sizes of all three parts are fixed. In what follows, we refer to this buffer organization as "static allocation".

Partition boundaries are then determined by drawing samples from the larger of the two relations to be joined. The samples are used to estimate the distribution of data. Depending on the sample rate, a part of the outer relation buffer is reserved for potential overflow tuples. The actual sampling percentage is determined by estimating the cost for the join process. The optimal total join cost is a trade-off between the join cost and the sampling cost.

Long-lived tuples have to be kept for each partition they overlap to find all possible join partners. To avoid physically duplicating these tuples in each relation, a certain part of the buffer is reserved for them, and the partitions must be visited in time order. After each join step, only tuples not valid in the next partition are removed. Long-lived tuples are held in buffer. Long-lived tuples of the outer relation are kept in the outer relation buffer, long-lived tuples of the inner relation are stored in the tuple cache page and flushed to disk if necessary.

4.2 The *Time-Index* join

The join algorithm of [RF93] makes use of the *Time Index* to compute a perfect partitioning, where each partition is guaranteed to fit into the allocated buffer space. The Time Index is an index structure for temporal data that stores pointers to tuples in order of indexing points. An indexing point is created when a version of an object starts or ends at this time point. Indexing points can be stored in a B^+-tree like structure. Tuple pointers can be split into three buckets for an indexing point: (i) tuples that started at that point, (ii) tuples ending at that point, and (iii) tuples still active from a previous indexing point. In an index node, only the first entry contains pointers to all active tuples. The other entries contain only incremental changes. Together with the pointers, the number of tuples in each bucket is maintained. For the join, each relation is allocated a fixed amount of buffer space (by default, half) and the partitioning process scans the time index to find the largest starting time for the next partition.

The paper [RF93] also considers joining append-only relations, and in this case advocates a partition-join with dynamic buffer allocation. The partitioning process estimates the arrival rate of new tuples for each relation using the *Time Index* and allocates buffer space to each relation based on this estimate, and then follows the partitioning process above. It is not clear why the authors restrict dynamic partitioning to append-only relations.

5 Partition-Join Using Histograms

Although the partition-based natural join has proved ([SSJ94]) to be more I/O cost efficient than the nested-loop and sort-merge-join, several aspects can still be

significantly improved. The I/O accesses for sampling can be completely avoided if information about the data distribution, such as a histogram, is already available in the system. For the Time-Index join, using an index not only involves additional block accesses during the join process but also requires building and maintaining an index on the join attribute in the system. We present a partition-based temporal join algorithm that computes an optimal dynamic partitioning using temporal histograms. This algorithm uses dynamic buffer allocation and does not require the data to be stored in any particular way.

5.1 Dynamic Buffer Allocation

Investigation of the cost of join algorithms has shown that algorithms should try to maximize the number of blocks read from disk for both relations instead of reading one relation in large parts and the other relation pagewise [HR96]. For the temporal histogram partition-join, we use this strategy, which also leads to better buffer utilization. The sizes of the input buffers of both relations vary in size from partition to partition depending on the ratio of active tuples in each relation in the current partition. The sizes of the input buffers are also determined during the partitioning computation step.

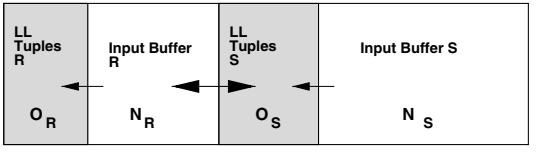


Fig. 2. Storage organization for the temporal partition-join

Figure 2 illustrates this memory organization. The buffer is divided into a smaller part for relation R and a larger part for relation S. Within the input buffers, a certain amount of space (marked grey in the figure) is occupied by long-lived tuples.

The histogram partition-join can also be executed using a static buffer allocation as described in Section 4.1.

5.2 Computing Partition Boundaries

As overflow of partitions leads to expensive additional disk accesses during the join, the quality of the partition boundaries is the crucial part for the performance of a partition-join. We use temporal histograms to calculate the partitioning.

As defined in Section 3, we assume we have histograms for the start times of tuples as well as for the number of active long-lived tuples. The algorithm scans through the time range of the relations. A partition is formed every time the number of blocks required to store the number of tuples still active and the number of tuples starting during the partition fills the available buffer space.

In the following algorithm, $Act_P(t)$ is the estimated number of tuples active for each relation P at time t. We can calculate $Act_P(t)$ from the histogram as

$$Act_P(t) = ACT_P[j] + START_P[j], \text{where } j = \max\{k \mid H_P[k] \leq t\}$$

We calculate $Start_P(t)$, the estimated number of tuples starting at time t, as

$$Start_P(t) = \begin{cases} START_P[j], & \exists j\ t = H_P[j] \\ \infty, & t = \mathsf{end\_time} \\ 0, & \text{otherwise} \end{cases}$$

Effectively, we treat each tuple in a histogram bucket i as starting at the start of that bucket ($H_P[i]$) and existing over the entire bucket time range.

B_{total} is the total buffer space (in blocks) available for the partitions of the two relations. The expression $block(n)$ is the number of blocks required to store n tuples. $PBounds[i]$ is the start time for the i^{th} partition. $T_R[i]$ and $T_S[i]$ denote the number of tuples in partition i calculated so far. $S_R[i]$ and $S_S[i]$ are the buffer sizes (in blocks) reserved for R and S in partition i. The algorithm makes no assumptions on the form (e.g. equi-width or equi-height) of the histograms, or that the histograms for R and S share common boundaries.

The time counter t is initialized to the earliest start time, and the number of partitions is set to zero. Until the whole of the time range of the relations has been covered, the algorithm tests if it can include the next time point in the current partition. If the tuples that start at this time point could be added to the partition without overflowing the buffer, the partition is extended to include this time point. Otherwise, the current partition is finished and a new partition is started. For the new partition, the tuple counter is again initialized with the number of long-lived tuples at the start of the partition.

```
calculate_partitioning(R,S)
  t := start_time; np:= 0;
  while t < end_time
    np++; PBounds[np] := t;
    T_R[np] := Act_R(t); T_S[np] := Act_S(t);
    t := t + 1;
    while (Start_R(t) = 0 and Start_S(t) = 0) or
          (block(T_R[np] + Start_R(t))) + (block(T_S[np] + Start_S(t))) < B_Total)
      T_R[np] = T_R[np] + Start_R(t); T_S[np] = T_S[np] + Start_S(t);
      t := t + 1;
    S_R[np] = block(T_R[np]); S_S[np] = block(T_S[np]);
  return PBounds;
```

This algorithm determines the partition boundaries and buffer sizes for each partition taking into account the space for both long-lived tuples and newly starting tuples. The resulting partition boundaries are used to physically partition the two relations using Grace partitioning [KTM83]. Note that indirectly partition boundaries will always be histogram boundaries.

5.3 Joining the Partitions

The main join algorithm reads each partition and joins it in main memory using any kind of suitable join. Because long-lived tuples occur in more than one partition, the join is executed in three steps to avoid joining long-lived tuples again. For each partition, the tuples for each relation are read and joined with the other relation's active long-lived tuples (O_R, O_S), and with each other. After the join, the long-lived tuples for the next partition are determined and kept (in fact, with care, this can be folded into the join).

```
join_partitions(R,S)
  O_R := ∅; O_S := ∅;
  for i = 1 to np
    N_R := ReadPartition(R,i); N_S := ReadPartition(S,i);
    Join(N_R,O_S); Join(N_R,N_S); Join(O_R,N_S);
    O_R := KeepLonglivedTuples(N_R ∪ O_R);
    O_S := KeepLonglivedTuples(N_S ∪ O_S);
```

5.4 Overflow Treatment

As histograms are only an estimation of the distribution of data, partition boundaries can be miscalculated and this may cause the partition to overflow. Buffers

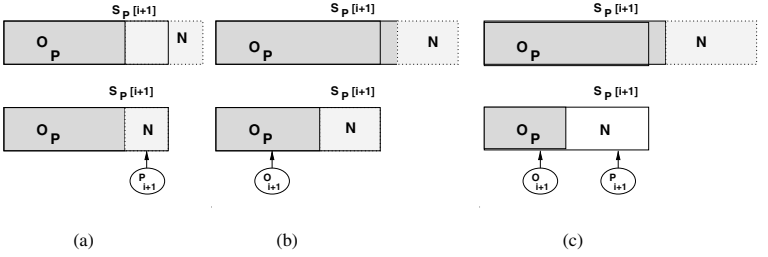


Fig. 3. Overflow treatment for the partition-join

are internally separated into blocks for long-lived tuples and new tuples. When we are about to read in a next partition $i+1$ for relation P consisting of N blocks, we have $S_P[i+1]$ blocks of allocated buffer space and $block(|O_P|)$ blocks already holding long-lived tuples for P. If $block(|O_P|) < S_P[i+1] < N + block(|O_P|)$ then we cannot fit all the long-lived tuples and new tuples in the available buffer space. In this case $S_P[i+1] - block(|O_P|)$ blocks are used for holding new tuples of partition $i+1$. During the join, we read partition $i+1$ in chunks of $S_P[i+1] - block(|O_P|)$ blocks (Figure 3 (a)).

If $block(|O_P|) > S_P[i+1]$ and $N \leq S_P[i+1]/2$, the long-lived tuples by themselves do not fit in the allocated buffer space. In this case, we keep $S_P[i+1] - N$ blocks of long-lived tuples in the buffer and save the remaining tuples to a separate overflow file, similar to the tuple cache in the static buffer allocation.

The entire partition of new tuples is read in, and during the join we reread the flushed long-lived tuples (Figure 3 (b)).

If $block(|O_P|) > S_P[i+1]$ and $N > S_P[i+1]/2$ then conceivably too many long-lived tuples would be flushed to disk, so we allocate half of the available buffer space ($S_P[i+1]/2$ blocks) to long-lived tuples and half to new tuples, and save excess long-lived tuples to disk before performing the join (Figure 3 (c)).

6 Experimental Evaluation

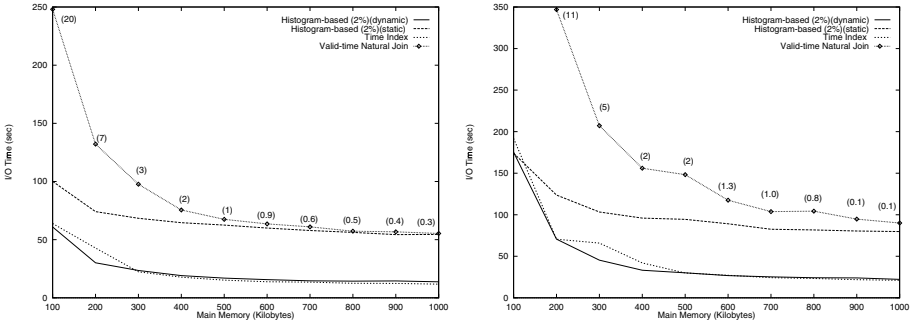
We created testdata with a uniform and a normal distribution of its active tuples. The size of the uniformly distributed relation is 50,000 tuples. The normally distributed relation contains 99,000 tuples. We used a blocksize of 4 Kbytes in the experiments. The size of a tuple is 128 bytes. To create histograms, 2 per cent of the tuples were used as samples (by default). We used total histograms (i.e. histograms with width 1) in the experiments (by default). More details can be found in [SS00].

We assume a random access every time a sequence of blocks is written or read from file. Each of these operations involves one random access followed by the number of sequential accesses. The average disk seek time used is $TS = 10$ms, while the transfer time is $TT = 1$ms/block. The ratio of seek to transfer time is conservative with respect to modern disk drives. All the experiments compare the I/O costs for processing the input, and do not include I/O costs for writing the result of the join. We have not included I/O for handling histograms which is completely overshadowed by the join cost [SS00].

Sensitivity to main memory buffer size: We first compare the performance of the histogram partition-join to the valid-time natural join and the Time Index. The Time Index join has been implemented using dynamic buffer allocation. We varied the size of the available buffer space from 100 Kbytes to 1000 Kbytes. We also measured the effect of using the static buffer allocation used by the valid-time natural join but computed the partitions using histograms. Figure 4 shows the input I/O times for the three algorithms for both distributions. The graph for the valid-time natural join is labeled with the percentage of tuples used for samples. Note that the costs for the valid-time natural join and histogram join do not include the I/O costs for sampling or reading histograms. The sampling time can be quite considerable. The costs for the Time-Index join include the index access costs during the partitioning process.

Both experiments show that the histogram partition-join using dynamic buffer allocation improves significantly on the valid-time natural join. Compared with the Time-Index join, the histogram partition-join demonstrates better performance for small buffer sizes and performance is approximately the same for large buffer sizes.

Static buffer allocation decreases performance significantly. I/O costs for the valid-time natural join are higher because of a less precise partitioning and a generally larger number of partitions due to the part of the buffer reserved for overflow. The Time-Index joins computed perfect partitions in most experiments



(a) Normal distribution (b) Uniform distribution

Fig. 4. Sensitivity to buffer size

and avoided overflow almost completely, but the positive effect is reduced by additional block accesses to the Time-Index during the partitioning process. We also compared the partition-join with histograms of higher accuracy (> 40 per cent sample rate) to the Time-Index join. Using more accurate histograms, the histogram partition-join consistently improved on the Time-Index join by about 10 per cent.

Necessary number of samples: We also investigated the effect of histogram accuracy on the performance. Table 1 shows the I/O costs for sample rates ranging from 0.01 per cent to 100 per cent. The buffer size used in both experiments is 500 Kbytes.

Sample	Uniform Distribution				Normal Distribution			
Rate	HistoJ	HistoJ	Time	PBVT	HistoJ	HistoJ	Time	PBVT
(in %)	Dynamic	Static	Index	Join	Dynamic	Static	Index	Join
0.01	85.48	244.70	15.21	63.53	62.04	301.26	29.94	117.47
0.10	30.18	66.60	15.21	63.53	33.14	96.18	29.94	117.47
1.00	16.93	62.99	15.21	63.53	30.04	91.29	29.94	117.47
10.00	16.79	59.87	15.21	63.53	30.09	89.80	29.94	117.47
100.00	13.97	55.75	15.21	63.53	26.24	83.22	29.94	117.47

Table 1. Sensitivity to histogram accuracy

The results show that histograms based on more than 0.1 per cent of tuples are still accurate enough to avoid significant overflow. Our experiments also showed that for the uniform distribution a sample rate of less than 40 per cent, for the normal distribution of less than 15 per cent, is enough to provide accurate enough histogram information to compute a perfect partitioning and achieve better performance than the Time-Index join. As the maintenance cost of histograms is extremely low compared to maintaining an index, this demonstrates that the histogram join offers better performance at a lower overall cost than the Time-Index join.

Effect of long-lived tuples: In a third experiment, we investigated the effect of a high percentage of long-lived tuples on the performance of the algorithms. Our experiments showed that a higher percentage of long-lived tuples has only little effect on the histogram partition-join. For details refer to [SS00].

Effect of histogram width: In the experiments, we used total histograms, i.e. histograms of width 1 chronon. We also examined the effect of histograms with wider buckets. Our experiments showed that histograms with greater width still provide accurate enough information to calculate a good partitioning. It appears that as long as histograms bars are not larger than partitions, they still provide enough useful information for a good partitioning.

Summary: Overall, our experiments showed that the histogram-based partition-join improves significantly on existing temporal partition-joins by demonstrating better performance than the valid-time natural join. Without requiring an index and thus saving maintainance cost in the system, the histogram-based partition-join shows better performance than the Time-Index join given the accuracy of the histogram information is above a certain threshold and similar performance otherwise.

Acknowledgements We thank Kotagiri Ramamohanarao for his inspiration and support.

References

[GS91] H. Gunadhi and A. Segev. Query Processing Algorithms for Temporal Intersection Joins. In *Proc. 7th ICDE*, pages 336–344, April 1991.

[HR96] E.P. Harris and K. Ramamohanarao. Join Algorithm Costs Revisited. *The VLDB Journal*, 5(1):64–84, 1996.

[KTM83] M. Kitsuregawa, H. Tanaka, and T. Motoka. Application of hash to database machine and its architecture. *New Generation Computing*, 1(1):63–74, 1983.

[LM92] T.Y. Leung and R. Muntz. Temporal query processing amd optimization in multiprocessor database machines. In *Proc. of the 18th VLDB*, pages 383–394, 1992.

[PIHS96] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proc. of ACM SIGMOD Conf.*, pages 294–305, June 1996.

[PJ99] Dieter Pfoser and Christian S. Jensen. Incremental join of time-oriented data. In *Proc. of the 11th SSDBM*, pages 232–243, 1999.

[Poo97] Viswanath Poosala. *Histogram-based Estimation Techniques in Database Systems.* PhD thesis, University of Wisconsin – Madison, 1997.

[RF93] S. Rana and F. Fotouhi. Efficient Processing of Time-Joins in Temporal Data Bases. In *Proc. 3rd Int. Symp. on DB Systems for Advanced Applications*, pages 427–432, April 1993.

[SE96] D. Son and R. Elmasri. Efficient Temporal Join Processing Using Time Index. In *Proc. of the 8th SSDBM*, pages 252–261, June 1996.

[SS88] A. Segev and A. Shoshani. The Representation of a Temporal Data Model in the Relational Environment. In M. Rafanelli, J.C. Klensin, and P. Svensson, editors, *LCNS*, volume 339, pages 39–61. Springer-Verlag, 1988.

[SS00] Inga Sitzmann and Peter Stuckey. Histogram-based Temporal Joins. Technical Report TR2000/7, Department of Computer Science and Software Engineering, The University of Melbourne, 2000.

[SSJ94] M.D. Soo, R.T. Snodgrass, and C.S. Jensen. Efficient Evaluation of the Valid-Time Natural Join. In *Proc. of the 10th ICDE*, 1994.

Providing Topically Sorted Access to Subsequently Released Newspaper Editions or: How to Build Your Private Digital Library

Andreas Rauber and Dieter Merkl

Department of Software Technology, Vienna University of Technology
Favoritenstr. 9 - 11/188, A–1040 Vienna, Austria
www.ifs.tuwien.ac.at/∼andi, www.ifs.tuwien.ac.at/∼dieter

Abstract. Self-organizing maps are a popular neural network model for presenting high-dimensional input data on a two-dimensional map, providing a particularly useful interface to electronic document collections. However, as the size of the training data increases, both the necessary computational power as well as the training time required exceed tolerable limits. Still more important, not all training data may be available in one central location but may rather be collected and managed at different repositories or released in subsequent periods of time.
This paper describes an approach for combining independent, distributed self-organizing maps to build a higher order map, allowing the creation and maintenance of scalable, independent map systems, which can be built to suit the needs of individual users. This is achieved by training higher order maps using the trained lower order maps as input data. We demonstrate this approach by creating an integrated view of subsequent releases of a newspaper archive.

1 Introduction

During recent years we have witnessed the appearance of an ever increasing flood of miscellaneous written information available in computer accessible form, culminating in the advent of massive digital libraries. As an example of such a digital library consider the on-line archives of various newspapers and magazines. Powerful methods for organizing, exploring, and searching collections of text documents are thus needed to deal with that mass of information.

The map metaphor for displaying the contents of a document library in a two-dimensional display has gained increased interest in the information retrieval community [1, 4, 5, 7]. Maps are used to visualize the similarity between documents in terms of distance within the two-dimensional map display. Hence, similar documents may be found in neighboring parts of the map display similar to the topical organization of documents in a conventional library. One of the most prominent tools for automatically organizing documents by topic is an unsupervised neural network model, namely the self-organizing map (SOM). The underlying assumption in all of the above mentioned papers is that the data

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 499–508, 2000.

items, i.e. the documents, are collectively available at one site for SOM training. Moreover, most of the reports on the application of self-organizing maps for the organization of document libraries propose the usage of a single map for document space visualization. With ever growing information repositories, however, this assumption is questionable. One should rather expect that several repositories exist at different sites and a way for unified access should be provided for the users of such distributed document archives.

In this paper we suggest the utilization of individual self-organizing maps for representing several document archives such as different document collections or subsequently released editions of periodicals, e.g. annual collections. Multiple maps may then be integrated to produce an aggregated visualization of a larger portion of the document space. We demonstrate the effects of such an approach based on a sample library of text documents.

The remainder of this paper is organized as follows. In Section 2 we give a brief review of self-organizing maps and describe their integration. Section 3 is dedicated to a description of our experimental document library. Experimental results of our approach to document archive organization are given in Section 4. We provide an overview of related work in the application of unsupervised neural networks for document classification in Section 5. Finally, our conclusions are presented in Section 6.

2 Self-organizing maps

2.1 Architecture and training rule

The SOM [3] is a popular unsupervised neural network model, which abstracts from presented input data to provide a topology preserving mapping from a high-dimensional input space to a usually two-dimensional output space. It consists of a 2-dimensional grid of units with n-dimensional weight vectors matching the dimensionality of the input space. During the training process input signals are presented to the map in random order. An activation function is used to determine the winning unit as the unit showing the highest activation, where a common choice for such an activation function is the Euclidean distance. Next, the weight vectors of the winner and its neighboring units (as identified by a so-called neighborhood function) are modified following a time-decreasing learning rate to represent the input signal more closely. The result of this training process is a map providing a topology-preserving mapping in so far as similar input data are located close to each other on the 2-dimensional map display, i.e. similar input data are mapped onto neighboring units of the SOM. The weight vectors of the units, in turn, resemble as far as possible the input signals for which the respective unit serves as a winner.

2.2 Integration of independent maps

When using document representations as input data during the training process of self-organizing maps, each of the units represents a certain portion of

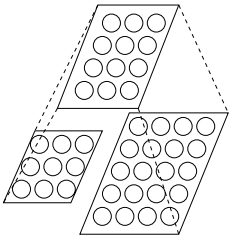


Fig. 1. Integration of two self-organizing maps

the document collection. Similar documents will be represented by neighboring units. In such a configuration it is necessary, that each document representation is stored locally at that site where the self-organizing map is trained. In case of a document archive that exists only distributed over several sites, it might be more efficient to have independent self-organizing maps that represent the various parts of the document archive than to transfer the whole information to one dedicated site for training. The increased efficiency relates to both reduced bandwidth and reduced computational load during the classification process because a smaller amount of documents are subject to classification. However, when some form of uniform access to the data is requested, the contents of the various sites has to be integrated.

With our approach to document archive organization we suggest to again use self-organizing maps to perform such an integration. In particular, the map that shall integrate different portions of the document archive may be trained by using the weight vectors of the maps to be integrated. Such a strategy may be applied recursively in order to build hierarchies of arbitrary depth as shown in Figure 1. In this figure a 3×3 and a 4×5 map are integrated to form a 3×4 map. Note, that also selected parts of self-organizing maps may be integrated by using essentially the same architecture. The user simply selects areas of interest scattered across different maps for which an integration shall be performed. By this, the user may tie together pieces of information to build her own library fine-tuned to her particular interests.

The effect of such an integration, obviously, is that similar input data items, that are separated in different low level maps are grouped together in the high level map. Input data that are mapped onto the same low level unit are represented together in the high level map.

3 An experimental document archive

For the experiments presented hereafter we use the classic *TIME Magazine* article collection[1]. It consists of a collection of 420 articles from the *TIME Magazine*

[1] available at *http://www.ifs.tuwien.ac.at/ifs/research/projects/somlib*

Set	#Articles	Articles	Dimension	Map Size
0	63	T000-T099	1433	6x10
1	85	T100-T199	1758	7x10
2	87	T200-T299	1812	7x10
3	72	T300-T399	2019	7x9
4	60	T400-T499	1761	6x9
5	53	T500-T599	1255	6x7
T	420	T000-T599	5923	10x15
I	359	weight vectors	3303	10x15

Table 1. Time Magazine Data - Experiment Setup

dating from the early 1960's. This collection, while being small enough to be presented in sufficient detail, provides the benefits of a real-world article collection covering a wide range of topics from foreign affairs to high-society gossip, thus forming an ideal testbed for the evaluation of our approach. To model a distributed library consisting of subsequent releases of a magazine, we split the document collection into 6 parts consisting of the documents T000 - T099, T100 - T199, ..., T500 - T599. Please note, that the consecutive numbering is not complete, i.e. not all articles are available in the package. Thus, we obtain 6 sets of documents of different size with each set containing between 53 and 87 documents. To allow convenient comparison, we also present results of using a standard single SOM for representing the complete data set *T*.

To be used for map training, a vector-space representation of the single documents is created. For each document collection a list of all words appearing in the respective collection is extracted while applying some basic word stemming techniques. Words that do not contribute to contents description are removed from these lists. Instead of defining language or content specific stop word lists, we rather discard terms that appear in more than 90% or in less than 3 articles in each collection. Thus, we end up with a vector dimensionality between 1255 and 2019 for the 6 document sets, cf. Table 1. The individual documents are then represented by feature vectors using a $tf \times idf$, i.e. term frequency $\times$ inverse document frequency, weighting scheme [11]. This weighting scheme assigns high values to terms that are 'important' as to describe and discriminate between the documents. These feature vectors are further used to train 6 self-organizing maps consisting of between 42 and 70 units. The weight vectors of these maps are then used to train an integrated SOM, forming data set *I*. An overview of the experimental setup is provided in Table 1.

4 Experimental results

4.1 The standard single SOM

In order to allow for a convenient comparison of the results obtained by integrating individually trained, independent SOMs we first present the results of the conventional approach of parsing all input data together to create one single

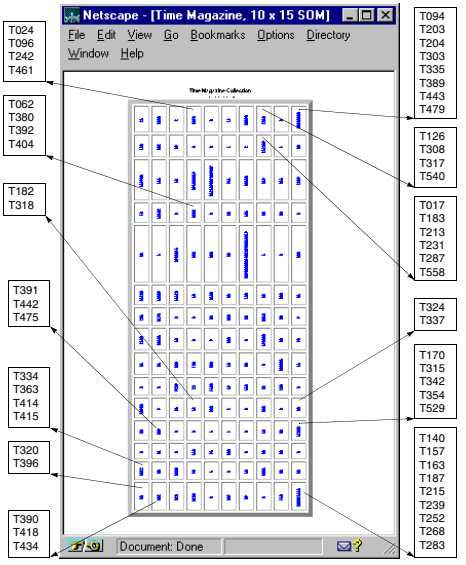


Fig. 2. 10 × 15 SOM of TIME Magazine Articles

set of input vectors to train a single flat SOM representing the whole data set. This results in a set of 420 input vectors of 5923 content terms. Using these as input to train a 10 × 15 SOM results in the document classification provided in Figure 2 where the documents are depicted as being mapped onto a grid of units. Due to space considerations we cannot present all the articles in the collection. We thus selected a number of units for detailed discussion, presenting the topics the respective documents are about.

We find, that the *SOM* has succeeded in creating a topology preserving representation of the topical clusters of articles. For example, in the lower left corner we find a group of units representing articles on the conflict in Vietnam. To name just a few, we find articles *T320, T369* on unit $(14/0)^2$, *T390, T418, T434* on unit (14/1) dealing with the government crackdown on Buddhist monks, next to a number of articles on units (14/3), (14/4) and neighboring ones, covering the fighting and suffering during the Vietnam War.

A cluster of documents covering affairs in the Middle-East is located in the lower right corner of the map around unit (14/9), next to a cluster on the so-called Profumo-Keeler affair, a political scandal in Great Britain in the 1960's, on and around units (10/9) and (11/9). Above this area, on units (6/10) and neighboring ones we find articles on elections in Italy and possible coalitions, next to two units (2/9) and (3/9) covering elections in India. Similarly, all other units on the map can be identified to represent a topical cluster of news articles.

[2] We use the notion (x/y) to refer to the unit located in row x and column y of the map, starting with (0/0) in the upper left corner

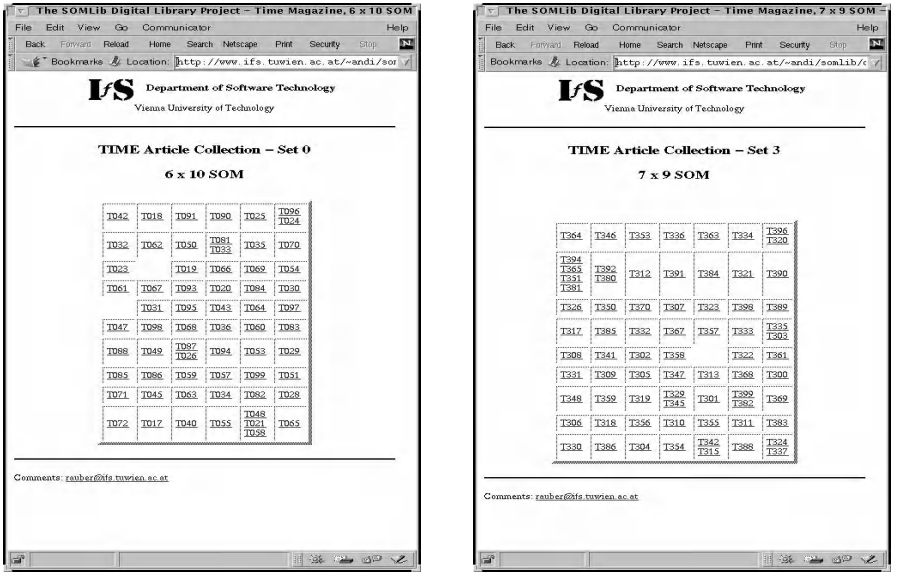


Fig. 3. 6 × 10 SOM of TIME Magazine Articles Sets 0 and 3

For a more detailed discussion of the articles and topic clusters found on this map, we refer to [10] and the online-version of this map with the respective articles, available at http://www.ifs.tuwien.ac.at/ifs/research/projects/somlib.

4.2 A distributed collection

In a distributed setting the feature vectors of the individual sets are used to train six self-organizing maps, each representing a subsection of the complete collection. As each map can and should be optimized for the requirements for each collection, we use differing map sizes for the six document sets to make up for the different number of documents in each set. For larger sets we thus train a map with more units than for smaller document collections, resulting in maps of between 42 and 70 units. An overview of the experimental setup is provided in Table 1, listing for each set the number of articles in the collection, the resulting feature space dimensionality, and the chosen map size.

Taking a closer look at the map representing set 0 in Figure 3 we find, for example, on unit (0/0) article *T042* entitled *"The View from Lenin Hills"* dealing with a discussion between Nikita Khrushchev and Soviet artists at the Lenin Hill Reception Palace, next to article *T018 - "Who's in Charge Here?"* about the failure of Khrushchev's virgin land plan for agriculture on unit (1/0) or *T032 - "Party Time"* on unit (0/1) on the New Year's Eve party at the Kremlin. On units (4/6), (5/6) and (5/7) we find the articles *T053, T029, T051* covering the war in Vietnam (Titles: *"Rice and Rats"; "The Helicopter War Runs into Trouble"; "The Strain of Constant Combat"*). Other groups of documents deal, for example, with the relation between India, Pakistan and China etc.

While each map thus represents a topically sorted organization of documents on the various topics, we find articles on most topics distributed across several sets, and thus across several maps. Let us consider, for example, documents on the situation in Vietnam and neighboring countries, which are spread across several maps. Apart from the documents on this topic found in map 0, we also have a number of documents on this topic on map 2 on units (3/0) (*T228 - "The Great Emancipator"*, *T269 - "The Pinprick War"*), (4/0) (*T202 - "A Wife is only a Wife, but Every Red is a Foe"*), and (5/0) (*T201 - "After the Party"*, *T226 - "The Undertaker"*, *T227 - "A New Civil War"*). Further articles on this topics are located in the upper right corner of map 3 (cf.Figure 3) and map 4, with articles (*T334, T396, T320, T390*) and (*T414, T418, T434, T415, T464, T498, T470, T480*) respectively; or units (6/6) and (6/7) in the lower right corner of map 5 (articles *T508, T518, T533, T545*).

In a second step we now want to integrate these first level maps to again create a map representing the whole TIME collection in one single map. However, this time we will use the weight vectors of the 6 individually trained first level SOMs as input to the second level SOM. Since the weight vectors created from the 6 sets differ, as can immediately be told from the different feature vector dimensions for the sets provided in Table 1, we first have to create a unique feature vector representation based on the features present in the individual sets. In this case this leads to a feature space of 3303 content terms, which are now used to represent all 420 articles as represented by the 359 units of the 6 maps, which are now used as input vectors to create the integrating map.

To be able to compare with the single flat SOM we make the integrating map the same size of 10×15 units. The resulting integrating SOM given in Figure 4, which, instead of representing the document vectors on its units, lists the units of the 6 individual SOMs, which in turn represent the corresponding articles. Again, due to space considerations, we can only present a subset of all mapped documents in detail, in this case via their representative weight vectors of the individual units. Thus, in this figure we find, instead of document numbers, a pointer to the relevant map m followed by the *id* of the mapped unit of that map as (x/y), again identifying the location of the unit in the respective map by column x and row y. The pointer *1_(2/3)* thus identifies the unit in column 2, row 3 on map 1.

For the integration process to be feasible, the resulting clusters on both the integrated and the standard flat SOM should be similar. When evaluating the two SOMs, we find the topology preserving mapping and clustering of the integrating SOM to be comparable to the one that was trained directly using the document description vectors. The cluster on *Vietnam* is now located in the upper right area of the integrating map. For example, we find on unit (9/3) 4 units from 2 different maps, namely the units (5/0), (6/0) and (6/1) from map 3 and unit (4/0) from map 4. These, in turn, represent a number of articles that are already familiar to us, namely *T320, T334, T390, T396, T414* and *T418*. All of these articles deal with the religious conflict in Vietnam. Looking at the neighboring unit above, i.e. (9/2), we find 3 units from map 4, namely (5/0),

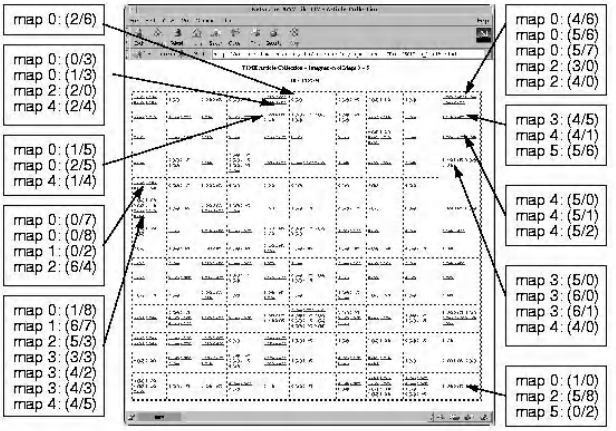


Fig. 4. 10 × 15 SOM integrating 6 independently trained SOMs

(5/1) and (5/2), representing articles *T434, T480* and *T498*. All of them are part of this cluster on the religious conflict too. However, there are two more documents in this cluster on the single SOM presented there, namely articles *T415* and *T363*, located on unit (0/14) in the single flat SOM. Since they were considered part of that topical cluster on the single, large SOM representing all the articles directly, we should expect them to be located in the same cluster on the integrating map. This is actually the case, since the two documents, located on units (4/0) on map 3 and (3/1) on map 4, are right next to the the other two units in this cluster on the integrating map, namely on units (8/3) and (8/4) of the integrating map. We thus find the cluster on the religious conflicts to consist of 4 units in the integrating map, which happens to be identical to the number of units in the original single map.

As part of the large Vietnam cluster, next to the articles on the religious conflicts in the single map we found the cluster on the war and the fighting in Vietnam. The same is the case in the integrating map, where, for example, the next unit up the column, i.e. (9/1), represents 3 units from 3 different maps, which in turn represent a set of articles, namely on unit (4/5) of map 3 article *T313*; on unit (4/1) of map 4 article *T464*; and on unit (5/6) of map 5 articles *T518, T533* and *T545*. We thus find this unit to represent sections from 3 different library maps, containing a total of 5 articles. All of these articles are part of the cluster on the fighting in Vietnam. The same is the case for the neighboring corner unit (9/0), which represents a total of 5 units from 2 different maps and a total of 6 articles, and its neighboring units (8/0) and (8/1). We thus find the cluster of articles on Vietnam to span about the same number of units in the integrating second level map as in the single SOM. Other document clusters identified on the individual maps can be found, such as the Profumo-Keeler scandal located around unit (0/12) or the elections in Italy around units (2/5).

Thus, while the orientation of the integrating map is drastically different with Vietnam, for example, now being located in the upper right corner of the

map, rather than on the lower left corner as in the single first level SOM, the organization of the articles on the map, and thus the topical clusters are more or less identical. This makes the integration approach an interesting and feasible alternative to the repeated training of huge library maps.

5 Related work

Text classification with self-organizing maps already has produced an impressive record of publications. The work of [4] perhaps marks the first attempt to use the self-organizing map in an information retrieval setting. In this work the authors report the results from classifying a number of technical documents based on keywords extracted from the various titles. In total, the document representation is made up from 25 manually selected index terms and is thus not really realistic. The results, however, showed that the self-organizing map was highly effective in representing the similarities of the various documents. This line of research is continued in [5], yet this time with full text indexed documents and thus by using a more realistic high-dimensional document representation.

Apart from the self-organizing maps a number of other artificial neural network architectures have been used for document classification. In [6] a comparison between adaptive resonance theory and self-organizing maps is provided based on their results in document classification. As a severe shortcoming of adaptive resonance theory in this kind of application we have to note the lack of intercluster similarity representation. In other words, the information concerning the similarity of documents assigned to different classes by the training process cannot be deduced from the training result. Growing cell structures, a neural network with an adaptive architecture that grows and splits during the training process according to the requirements of the input space, are used in [2] for document classification. In principle, this model seems to be feasible for document classification in that intracluster similarities are represented in a map-like display. Intercluster similarity, however, cannot be represented. Additionally, the neural network model requires more training parameters, related to network structure adaptation, that are to be adjusted prior to the training process, and the training results are more susceptible to minor variations in these parameters. Only recently a series of experiments using *hierarchical feature maps* [9] for document archive organization has been described [7, 8]. This neural network model allows hierarchical document clustering relying on a predefined hierarchical neural network architecture. The building-blocks of *hierarchical feature maps* are independent self-organizing maps. The benefits are related to, first, dramatically increased training speed, and, second, an intuitive representation of document similarity even in large archives. The major shortcoming, obviously, is the need for defining the hierarchical network architecture prior to training which requires some knowledge of the particular characteristics of the document archive to be organized.

6 Conclusions

In this paper we have presented a novel approach for organizing distributed document archives. In our approach different portions of a document archive that may reside on different sites are managed independently by means of self-organizing maps. These maps may be integrated in order to provide an overview of the information contained in the document archive. We suggested that the integration of low level maps should be based on the weight vectors representing groups of documents. This results in reduced computational load during the integration as compared to an equally well applicable approach where the various documents are used directly.

The example discussed in this paper is based on the integration of complete low level maps. The integration phase, however, is not restricted to complete maps. The user may equally well define areas of interest that cover just portions of low level maps. These areas can be integrated by using the very same integration process. The benefit for the user of such a distributed document archive is that she may define her own personal library that reflects her particular interests.

References

1. S. Kaski, T. Honkela, K. Lagus, and T. Kohonen. Creating an order in digital libraries with self-organizing maps. In *Proc. of the World Congress on Neural Networks*, San Diego, CA, 1996.
2. M. Köhle and D. Merkl. Visualizing similarities in high dimensional input spaces with a growing and splitting neural network. In *Proc. Intl. Conf. on Artificial Neural Networks*, Bochum, Germany, 1996.
3. T. Kohonen. *Self-organizing maps.* Springer-Verlag, Berlin, 1995.
4. X. Lin, D. Soergel, and G. Marchionini. A self-organizing semantic map for information retrieval. In *Proc. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Chicago, IL, 1991.
5. D. Merkl. A connectionist view on document classification. In *Proc. Australasian Database Conf.*, Adelaide, Australia, 1995.
6. D. Merkl. Content-based software classification by self-organization. In *Proc. IEEE Intl. Conf. on Neural Networks*, Perth, Australia, 1995.
7. D. Merkl. Exploration of text collections with hierarchical feature maps. In *Proc. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Philadelphia, PA, 1997.
8. D. Merkl and A. Rauber. CIA's view of the world and what neural networks learn from it: A comparison of geographical document space representation metaphors. In *Proc. Int'l Conf. on Database and Expert Systems Appl.*, Vienna, Austria, 1998.
9. R. Miikkulainen. Script recognition with hierarchical feature maps. *Connection Science*, 2, 1990.
10. A. Rauber and D. Merkl. Using self-organizing maps to organize document collections and to characterize subject matters: How to make a map tell the news of the world. In *Proc. 10th Int'l Conf. on Database and Expert Systems App. (DEXA99)*, Florence, Italy, 1999.
11. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.

Cost Estimation for Large Queries via Fractional Analysis and Probabilistic Approach in Dynamic Multidatabase Environments*

Qiang Zhu, S. Motheramgari, and Yu Sun

Department of Computer and Information Science,
The University of Michigan, Dearborn, MI 48128, U.S.A.
{qzhu, motheram, yusun}@umich.edu

Abstract. Research on query cost estimation for local database systems in a multidatabase system (MDBS) has attracted many researchers in the database area recently. Obtaining good query cost estimates is crucial for performing effective global query optimization in an MDBS. However, most techniques suggested so far, including database calibrating and query sampling, consider only a static multidatabase environment. Recently, we proposed a qualitative approach to developing cost models for a dynamic multidatabase environment. It has been shown that this approach is promising in estimating the cost of a query run in any given contention state for a dynamic environment. However, a large (cost) query in practice may experience multiple contention states during its execution, which cannot be directly handled by the qualitative approach. In this paper, we propose two new techniques, i.e., fractional analysis and probabilistic approach, to solve the problem. The fractional analysis technique, which is suitable for a system environment that changes contention states gradually and smoothly, estimates a query cost by analyzing its fractions. The probabilistic approach, which is suitable for a system environment that changes contention states rapidly and randomly, estimates a query cost based on the theory of Markov chains. Cost estimation formulas for both techniques are derived, and their properties are studied. Our experimental results demonstrate that the suggested techniques are quite promising in estimating costs for large queries in a dynamic multidatabase environment.

1 Introduction

To meet users' increasing needs to access data from multiple pre-existing databases managed by heterogeneous database management systems (DBMS), multidatabase systems (MDBS) have been studied by many database researchers in recent years[7–9, 13]. An MDBS is a global system built on top of multiple local (component) DBMSs and provides users with a uniform interface to access local

* Research supported by the US National Science Foundation under Grant # IIS-9811980 and The University of Michigan under OVPR and UMD grants.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 509–525, 2000.

databases. A key feature of an MDBS is local autonomy that each local database system retains to serve existing local applications. The global system can only interact with local DBMSs at their external user interfaces.

A global query issued on an MDBS is decomposed into a set of local queries executed at local database systems during query processing. The results from local queries are integrated into the final query result returned to its user. However, the way to decompose a global query is not unique. Different decomposition strategies may yield significantly different performance in the distributed environment. Choosing a good decomposition and integration strategy for a given global query is the task of global query optimization. To perform global query optimization, cost information for local queries to be performed on local database systems is required. However, such information is unavailable to the global query optimizer since the internal implementation details of a local DBMS is unknown to the MDBS. Estimating the costs of local queries at the global level in an MDBS is a major challenge for global query optimization in the system.

To tackle this challenge, a number of techniques have been proposed in the literature. In [3], Du *et al.* proposed a calibration method that makes use of the observed costs of some special queries run against a special synthetic calibrating database to deduce necessary local cost parameters. In [6], Gardarin *et al.* extended Du *et al.*'s method so as to calibrate cost models for object-oriented local database systems in an MDBS. In [15–17], Zhu and Larson proposed a query sampling method that develops regression cost models for local query classes based on observed costs of sample queries run against actual user databases. In [14], Zhu and Larson introduced a fuzzy method based on fuzzy set theory to derive fuzzy cost models in an MDBS. In [10], Naacke *et al.* suggested an approach to combining a generic cost model with specific cost information exported by wrappers for local database systems. In [1], Adali *et al.* suggested to maintain a cost vector database to record cost information for every query issued to a local database system. In [12], Roth *et al.* introduced a framework for costing in the *Garlic* federated system.

All above techniques considered only a static environment, i.e., assuming that the environment does not change significantly over time. However, such an assumption may not be true in reality since many factors such as the number of concurrent processes in a multidatabase system environment may change significantly. The cost of a query can be dramatically different at different times in a dynamic system environment. For example, in one of our experiments, the cost of a sample query[1] performed on Oracle 8 in a dynamical environment varied from 2.58 sec. to 127.05 sec. (49 times!) when we had 1 to 30 concurrent user processes in the environment. Hence query cost estimates obtained for a static environment cannot be used in a dynamic environment.

To capture dynamic factors in a cost model, we recently proposed a qualitative approach[18]. This approach extends our previous query sampling method[15–17] and develops regression cost models using qualitative variables to indicate

[1] The query was *SELECT a1, a5, a7 FROM R WHERE a3 > 300 and a8 < 2000* on table $R(a1, a2, ..., a9)$ with 50,000 tuples of random data.

system contention states. Each contention state reflects a combined effect of dynamic factors on the system. Although such a cost model can be used to estimate the costs of queries for any contention state in the dynamic environment, each query is assumed to be run in a single contention state. The qualitative approach cannot directly solve the problem to estimate the cost of a large query run in multiple contention states.

To estimate the costs of large queries experiencing multiple contention states in a dynamic multidatabase environment, we develop two new techniques in this paper. The first technique, called fractional analysis approach, is to estimate query costs in a dynamic environment in which the system contention states change gradually and smoothly. The idea is to analyze and integrate the fractions of a query cost for multiple experienced contention states. The second technique, called probabilistic approach, is to estimate query costs in a dynamic environment in which the system contention states change rapidly and randomly. The idea is to make use of the theory of Markov chains to derive a cost formula to estimate the query costs in such an environment. These two techniques together with our qualitative approach provide a suite of techniques to estimate the costs of queries for different cases in a dynamic multidatabase environment.

The rest of this paper is organized as follows. Section 2 outlines our qualitative approach to developing cost models with qualitative variables for dynamic multidatabase environments. Section 3 presents the fractional analysis approach to estimating query costs. Section 4 discusses the probabilistic approach to estimating query costs. Section 5 shows some experimental results. The last section summarizes the conclusions.

2 Dynamic Cost Models with Qualitative Variables

To incorporate the dynamic factors in a multidatabase system into a cost model, we proposed an effective qualitative approach in [18]. In this approach, we consider the combined effect of all the factors on a query cost together rather than individually. Although the dynamic factors change differently in terms of changing frequency and level, they all contribute to the contention level of the underlying system environment, which represents their net effect. Notice that the cost of a query increases as the contention level. The system contention level can be divided into a number of discrete states (categories) such as "*High Contention*" (S_H), "*Medium Contention*"(S_M), "*Low Contention*" (S_L), and "*No Contention*" (S_N). A qualitative variable is used to indicate the contention states. This qualitative variable, therefore, reflects the combined effect of the dynamic environmental factors. A cost model including such a qualitative variable can capture the dynamic factors to certain degree.

Since, for a given query, its cost increases as the system contention level, we can use the cost of a probing query to gauge the contention level and classify the contention states for the dynamic system environment. To obtain an appropriate classification of system contention states, we first partition the range of a probing query cost in the given dynamic environment into subranges (inter-

vals) with an equal size. Each subrange represents a contention state. If some neighbor contention states are found to have a similar effect on the derived cost model, they are merged into one state. Such a uniform partition with merging adjustment procedure for a classification of contention states has been proven to be very effective in practice[18].

A qualitative variable X with M possible system contention states S_1, S_2, ..., S_M can be represented by a set of $M-1$ indicator (binary) variables Z_1, Z_2, ..., Z_{M-1}. That is, $X = S_i$ $(1 \leq i \leq M-1)$ is represented by $Z_i = 1$ and $Z_j = 0$ (for any $j \neq i$); and $X = S_M$ is represented by $Z_k = 0$ (for any $1 \leq k \leq M$). Including qualitative variable X in a cost model is equivalent to including indicator variables $Z_1, Z_2, ..., Z_M$ in the cost model.

To develop a cost model including the indicator variables, we extend our previous query sampling method in [15–17]. In other words, we use observed costs of sample queries to build a regression cost model with indicator variables as follows:

$$Y = \underbrace{(B_0^0 + \sum_{j=1}^{M-1} B_0^j Z_j)}_{intercepts} + \sum_{i=1}^{n} \underbrace{(B_i^0 + \sum_{j=1}^{M-1} B_i^j Z_j)}_{slopes} X_i, \qquad (1)$$

where Y is the query cost, X_i's are explanatory variables, Z_j's are indicator variables, and B_i^j's are the regression coefficients. The intercepts and slopes of equation (1) change from one contention state to another, indicated by the values of Z_i's. Since the above qualitative approach is obtained by introducing multiple contention states into our previous query sampling method, it is also called as the multi-states query sampling method. For more details of this method, please refer to [18].

3 Fractional Analysis Approach

One assumption made by the qualitative approach discussed in the last section is that the contention state does not change during the execution of a query although different executions of queries can be run in different contention states. This assumption is usually valid for small (cost) queries. For large (cost) queries, they may experience multiple contention states during their executions. How to estimate the cost of a query when it experiences multiple states during its execution is the issue to be discussed in this and the following sections.

There are two simple approaches to estimating the cost of a query experiencing multiple states. One is called the *single state analysis*. The idea is to ignore the changes in contention states during the execution of a query and use a dynamic cost model with a qualitative variable discussed in Section 2 together with one prevailing contention state to estimate the query cost. The prevailing contention state can be (1) the initial state in which the query is to start; (2) the median state among all states; or (3) a random state from all states. Unlike the initial state, the median and random states may not actually be experienced by the query at all. Hence the initial state may be superior in most cases for

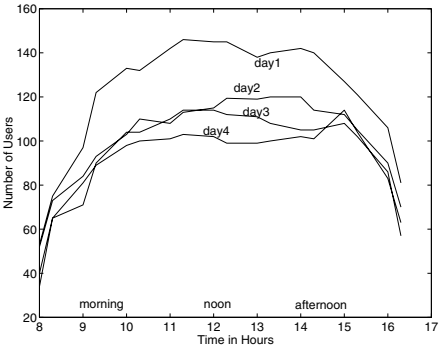


Fig. 1. System Loads in a Company

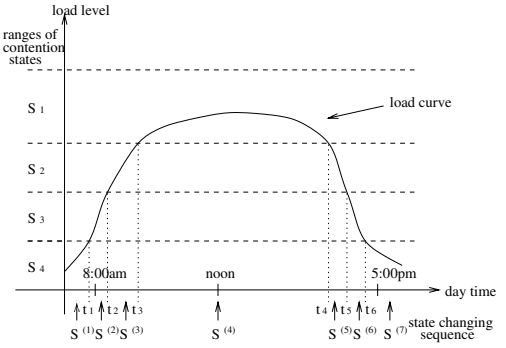


Fig. 2. A Typical Load Curve

the single state analysis approach. The advantage of this approach is that one step application of the dynamic cost model is sufficient to give a cost estimate. However, the resulting estimate may be inaccurate since not all experienced contention states are considered.

Another simple approach is called the *average cost analysis*. The idea is to take the average of costs for all the states in the environment as the cost estimate $C(Q)$ of query Q, that is, using $\bar{C}(Q) = \sum_{i=1}^{M} C(Q, S_i)/M$ to estimate $C(Q)$, where $C(Q, S_i)$ denotes the cost estimate for Q in state S_i and $S_1, S_2, ..., S_M$ are all possible states in a given environment. Although the average cost estimate is usually better than the single-state cost estimates, it may still be quite rough due to the fact that some contention states may never be experienced while other contention states may be experienced with various durations for the given query.

In this section, we are going to introduce a better cost estimation via a finer analysis, called *fractional analysis*. The key idea is to analyze a query cost by fractionalizing it according to the contention states to be experienced.

We notice that the system load in a particular application environment often demonstrates certain pattern. Fig. 1 shows the load for a system environment observed in a real-world company on different days. Clearly, the loads follow a similar pattern during every observed day in the company. The loads are minimum off working hours. The loads start to grow in the morning when the working hours begin and decline when the working hours are close to the end of the day. A curve depicting such a pattern in which the system load changes over time in an application environment[2] is called a *load curve*. Such a load curve can be obtained via calibrating the application environment under consideration. One assumption made in the following discussion is that the load curve for the given application environment is prior known.

As suggested in Section 2, the load (contention) level is divided into a number of discrete contention states (see Fig. 2), where the load level is measured by a probing query cost. Let $\Delta = \{ S_1, S_2, ..., S_M \}$ be the set of all possible contention

[2] In general, the time period for a load curve can be a day, a week, a year, or any other reasonable periodical durations. In this paper, we consider a day only.

states; $S^{(1)}, S^{(2)}, ..., S^{(N)}$ be the sequence of contention states occurred along the load curve in the given application environment, where $S^{(i)} \in \Delta$; $t^{(i-1)}$ and $t^{(i)}$ be the starting and ending times for state $S^{(i)}$ $(i = 1, 2, ..., N)$.

Consider query Q starting its execution at time $t_Q^{(s)}$ in state $S^{(k)}$. Let $C(Q, S^{(i)})$ $(i = k, k+1, ...)$ be the cost estimate for query Q if the query is executed entirely in state $S^{(i)}$.

If $C(Q, S^{(k)}) \leq (t^{(k)} - t_Q^{(s)})$, Q is expected to experience only one contention state $S^{(k)}$. Hence $C(Q, S^{(k)})$ is a good estimate of the cost for query Q, i.e., $C(Q) = C(Q, S^{(k)})$.

If $C(Q, S^{(k)}) > (t^{(k)} - t_Q^{(s)})$, query Q is expected to experience more than one contention state. Let $T^{(k)} = (t^{(k)} - t_Q^{(s)})$. Then $T^{(k)}/C(Q, S^{(k)})$ is the estimated fraction of work done for Q in state $S^{(k)}$. The remaining fraction $[1 - T^{(k)}/C(Q, S^{(k)})]$ of work for Q is to be done in the subsequent contention states. If $[1 - T^{(k)}/C(Q, S^{(k)})] * C(Q, S^{(k+1)}) \leq (t^{(k+1)} - t^{(k)})$, all remaining work of Q can be done in state $S^{(k+1)}$. The cost of Q can be estimated as: $C(Q) = T^{(k)} + [1 - T^{(k)}/C(Q, S^{(k)})] * C(Q, S^{(k+1)})$.

If $[1 - T^{(k)}/C(Q, S^{(k)})] * C(Q, S^{(k+1)}) > (t^{(k+1)} - t^{(k)})$, query Q is expected to experience more than two contention states. Let $T^{(k+1)} = (t^{(k+1)} - t^{(k)})$. Then $T^{(k+1)}/C(Q, S^{(k+1)})$ is the estimated fraction of work done for Q in state $S^{(k+1)}$, and $T^{(k)}/C(Q, S^{(k)}) + T^{(k+1)}/C(Q, S^{(k+1)})$ is the estimated fraction of work done so far (in both states $S^{(k)}$ and $S^{(k+1)}$). The remaining fraction $[1 - T^{(k)}/C(Q, S^{(k)}) - T^{(k+1)}/C(Q, S^{(k+1)})]$ of work for Q is to be done in the subsequent contention states. If $[1 - T^{(k)}/C(Q, S^{(k)}) - T^{(k+1)}/C(Q, S^{(k+1)})] * C(Q, S^{(k+2)}) \leq (t^{(k+2)} - t^{(k+1)})$, all remaining work of Q can be done in state $S^{(k+2)}$. The cost of Q can be estimated as: $C(Q) = T^{(k)} + T^{(k+1)} + [1 - T^{(k)}/C(Q, S^{(k)}) - T^{(k+1)}/C(Q, S^{(k+1)})] * C(Q, S^{(k+2)})$.

In general,

$$C(Q) = \sum_{i=k}^{m} T^{(i)} + [1 - \sum_{i=k}^{m} T^{(i)}/C(Q, S^{(i)})] * C(Q, S^{(m+1)}), \tag{2}$$

where $T^{(k)} = (t^{(k)} - t_Q^{(s)})$; $T^{(i)} = (t^{(i)} - t^{(i-1)})$ for $i \geq k+1$; m is the minimum integer such that $[1 - \sum_{i=k}^{m} T^{(i)}/C(Q, S^{(i)})] * C(Q, S^{(m+1)}) \leq T^{(m+1)}$.

Note that m cannot be determined in advance. It has to be determined during the fractional analysis. The following algorithm describes the fractional analysis procedure:

Algorithm 1 : Fractional Analysis

Input: The load curve including the contention states changing sequence $S^{(1)}$, $S^{(2)}$, ..., $S^{(N)}$ and the starting time $t^{(i-1)}$ and ending time $t^{(i)}$ for each state $S^{(i)}$ $(i = 1, 2, ..., N)$; the starting time $t_Q^{(s)}$ of query Q; the cost model $C(Q, S)$ for estimating the cost of query Q in any state S.
Output: Cost estimate $C(Q)$ for query Q.
Method:

1. **begin**
2. Find the initial state $S^{(k)}$ for Q such that $t^{(k-1)} \leq t_Q^{(s)} < t^{(k)}$;
3. Let $F := 0$; $C := 0$; $T := t^{(k)} - t_Q^{(s)}$; $m := k - 1$;
4. **while** $(1 - F) * C(Q, S^{(m+1)}) > T$ **do**
5. $C := C + T$;
6. $F := F + T/C(Q, S^{(m+1)})$;
7. $m := m + 1$;
8. $T := t^{(m+1)} - t^{(m)}$;
9. **end**;
10. $C := C + (1 - F) * C(Q, S^{(m+1)})$;
11. **return** C;
12. **end.**

Comparing the above fractional analysis with the single state analysis, we notice that, when query Q is expected to complete its execution entirely in its initial state (i.e., $C(Q, S^{(k)}) \leq (t^{(k)} - t_Q^{(s)})$), the cost estimates for Q from the fractional analysis and the initial single state analysis are identical. If such an initial state also happens to be the median (or randomly-selected) state, the query cost estimates from the fractional analysis and the median (or random) single state analysis are identical. However, in general, the execution of a large query may experience more than one contention state. Since the fractional analysis considers all the states that a query experiences, it usually gives better cost estimates than a single state analysis.

Comparing the fractional analysis with the average cost analysis, we have the following propositions:

Proposition 1. *Let $S^{(k)}, S^{(k+1)}, ..., S^{(m)}, S^{(m+1)}$ be the sequence of contention states experienced by query Q. Let I_j be the set of all indexes u's such that $S^{(u)}$ is in the sequence and $S^{(u)} = S_j \in \Delta$ for $j \in \{1, 2, ..., M\}$. Let $T_j = \sum_{u \in I_j} T^{(u)}$ (i.e., the accumulated duration[3] for Q in state S_j). If $T_1/C(Q, S_1) = T_2/C(Q, S_2) = ... = T_M/C(Q, S_M) = 1/M$ (i.e., $T_j = C(Q, S_j)/M$ for $1 \leq j \leq M$), then $C(Q) = \bar{C}(Q)$, where $C(Q)$ and $\bar{C}(Q)$ are the fractional cost estimate from (2) and the average cost estimate, respectively.*

Proof. Without loss of generality, we assume $S^{(m+1)} = S_M$. From (2), we have

$$\begin{aligned} C(Q) &= \sum_{j=1}^{M-1} \sum_{u \in I_j} T^{(u)} + \sum_{u \in I_M \ \wedge \ u \neq m+1} T^{(u)} + [1 - \sum_{j=1}^{M-1} \sum_{u \in I_j} T^{(u)}/C(Q, S_j) \\ &\quad - \sum_{u \in I_M \ \wedge \ u \neq m+1} T^{(u)}/C(Q, S_M)] * C(Q, S_M) \\ &= \sum_{j=1}^{M-1} T_j + T_M - T^{(m+1)} + [1 - \sum_{j=1}^{M-1} T_j/C(Q, S_j) - T_M/C(Q, S_M) \qquad (3) \\ &\quad + T^{(m+1)}/C(Q, S_M)] * C(Q, S_M) \end{aligned}$$

[3] Assume that $T_j = 0$ if $I_j = \emptyset$ (empty set).

$$= \sum_{j=1}^{M-1} C(Q,S_j)/M + C(Q,S_M)/M - T^{(m+1)} + [1-(M-1)/M - 1/M$$

$$+T^{(m+1)}/C(Q,S_M)] * C(Q,S_M) = \sum_{j=1}^{M} C(Q,S_j)/M = \bar{C}(Q).$$

Therefore, the cost estimate from (2) and the average cost estimate are identical in such a case. □

Note that $T_j/C(Q,S_j)$ is the fraction of work done for Q in state S_j (during its total stay in the state if there are several visits). Proposition 1 actually states that the cost estimate from the fractional analysis is identical to the average cost of all states when query Q experiences every possible contention state in Δ at least once and completes an equal fraction $(1/M)$ of work in each of the M states. In such a case, both cost estimates are quite accurate. However, in general, a query may finish more work in one state than others, which implies that the above condition does not hold. In this case, the fractional analysis is expected to give better estimates since it considers the actual fraction of work done in each state for the query.

Proposition 2. *Assume that $C(Q,S_j) \neq C(Q,S_i)$ for some $j \neq i$. Using the same notation as in Proposition 1, if $T_1 = T_2 = ... = T_M$, then $C(Q) < \bar{C}(Q)$.*

Proof. Let $T = T_j$ $(1 \leq j \leq M)$. Note that $T_j/C(Q,S_j)$ $(1 \leq j \leq M)$ is the fraction of work done for Q in state S_j. The fractions in all states should add to 1, that is: $\sum_{j=1}^{M} T_j/C(Q,S_j) = 1$. Hence, $T = 1/[\sum_{j=1}^{M} 1/C(Q,S_j)]$. From (3), we have

$$C(Q) = M/[\sum_{j=1}^{M} 1/C(Q,S_j)]. \tag{4}$$

On the other hand, we have

$$[\sum_{j=1}^{M} 1/C(Q,S_j)] * [\sum_{i=1}^{M} C(Q,S_i)] = \sum_{j=1}^{M}\sum_{i=1}^{M} C(Q,S_i)/C(Q,S_j)$$

$$= M + \sum_{j=1}^{M-1}\sum_{i=j+1}^{M} [C(Q,S_i)/C(Q,S_j) + C(Q,S_j)/C(Q,S_i)]. \tag{5}$$

Notice that $\frac{a}{b} + \frac{b}{a} \geq 2$ for $a,b > 0$ and the equality is true only if $a = b$. Since $C(Q,S_j) \neq C(Q,S_i)$ for some $j \neq i$ is assumed, i.e., $C(Q,S_i)/C(Q,S_j) + C(Q,S_j)/C(Q,S_i) > 2$ for some $j \neq i$. Hence, from (5)

$$\sum_{j=1}^{M} 1/C(Q,S_j) * \sum_{i=1}^{M} C(Q,S_i) > M + \sum_{j=1}^{M-1}\sum_{i=j+1}^{M} 2 = M^2. \tag{6}$$

From (4), (6) and the average cost estimate formula, we have

$$\frac{\bar{C}(Q)}{C(Q)} = [\sum_{j=1}^{M} 1/C(Q,S_j)] * [\sum_{i=1}^{M} C(Q,S_i)]/M^2 > \frac{M^2}{M^2} = 1.$$

Therefore, $C(Q) < \bar{C}(Q)$. □

People might think that a query cost would be equal to the average cost for all states if the query spends the same amount of time in every state. However, Proposition 2 states that the cost estimate for a query from the average cost analysis is larger than the cost estimate for the query from the fractional cost analysis when query Q spends an equal amount of time in every state. The reason for this phenomenon is that Q runs in different states with different working rates. The higher the contention level is, the slower the working rate. Therefore, with the same amount of time, the query will complete less work in a state with a higher contention level. If all states spend the same amount of time on Q, most work of Q will be done in the states with lower contention levels. The actual cost of Q will be smaller than the average cost in such a case.

4 Probabilistic Approach

Although the fractional analysis approach in the last section can estimate costs for queries experiencing multiple contention states during their executions, one assumption made is that the load curve in the given dynamic environment is prior known and the load changes gradually. To deal with the cases with rapidly and randomly changing loads in a dynamic environment, we develop a probabilistic approach in this section.

Note that a rapidly and randomly changing load in a dynamic environment causes frequent changes in its contention states. The occurrence of a contention state is a random phenomenon and governed by laws of probability.

Let $\Delta = \{ S_1, S_2, ..., S_M \}$ be the set of all possible contention states in a dynamic environment. We consider a sequence of occurrences of contention states $\{X^{(n)}, n = 0, 1, 2, ...\}$ in the given environment as a stochastic process, where $X^{(n)}$ is a random variable taking values from Δ. $X^{(n)} = S_i$ indicates that the environment is in contention state S_i at time $t_n = t_0 + n * \delta$, where δ is the observing time interval. We notice that the probability for the next contention state $X^{(n+1)}$ taking a particular value usually depends only on the value of the present contention state $X^{(n)}$ and is independent of values of past contention states $X^{(0)}, X^{(1)}, ..., X^{(n-1)}$. For example, if the present contention state is "very busy", the next contention state is most likely to be "quite busy" or "extremely busy" regardless of past contention states. In other words, the conditional distribution of any future state $X^{(n+1)}$ satisfies:

$$\begin{aligned} &P(X^{(n+1)} = S_j \mid X^{(n)} = S_i, X^{(n-1)} = S_{k_{n-1}}, ..., X^{(0)} = S_{k_0}) \\ &= \quad P(X^{(n+1)} = S_j \mid X^{(n)} = S_i) \quad = \quad P_{ij}, \\ &\qquad\qquad\qquad\qquad \textit{for any } n \geq 0, \textit{ and } S_j,\ S_i,\ S_{k_{n-1}},\ ...,\ S_{k_0} \in \Delta. \end{aligned}$$

P_{ij} $(1 \leq i, j \leq M)$ denotes the (one-step) transition probability for the system contention state changing from S_i to S_j in the next time interval. Clearly, $P_{ij} \geq 0$ and $\sum_{j=1}^{M} P_{ij} = 1$ since the system has to be in one of the states in Δ in the next time interval. Such a stochastic process is known as a (finite) Markov chain[11].

The next issue is how to establish the transition probabilities in the Markov chain for a dynamic environment. Note that the contention state in a dynamic environment after each time interval can either remain in the same state or change to other states. However, the probability for the contention state changing to a far-away state is less than the one for it changing to a neighbor state.

Let the system contention state at time t_0 be S_i, and the system contention state at next time t_1 be S_j. Recall that a contention state reflects a set of close contention levels which are measured by probing query costs. Let L_k $(1 \leq k \leq M)$ be the center of gravity of the contention levels for contention state $S_k \in \Delta$. Let d_{ij} be the distance between L_i and L_j.

If the probability for the system contention state remaining in the same state S_i is q_i, the probability for the system contention state changing to other states from S_i will be $(1 - q_i)$. Among other states, a reasonable assumption is that the probabilities are inversely proportional to their distances to S_i. Hence,

$$P_{ii} = q_i; \quad P_{ij} = [(1 - q_i)/d_{ij}]/[\sum_{j=1}^{M} 1/d_{ij}], \qquad for\ 1 \leq i, j \leq M.$$

Parameter q_i can be calibrated via experiments. Matrix $(P_{ij})_{M \times M}$ lists all one-step transition probabilities for the Markov chain.

The probability $P_{ij}(n)$ for a contention state S_i changing to another contention state S_j after n time intervals is called an n-step transition probability for the Markov chain. For a finite Markov chain, the limit $\pi_j = \lim_{n \to \infty} P_{ij}(n)$ exists[11] and is called the limit probability of state S_j. Two interesting properties[11] of a limit probability are: (1) it is independent of the initial state (i.e., S_i) and (2) it not only represents the probability of a contention state in a Markov chain after a sufficiently large number of transitions but also represents the long-run portion of time for the Markov chain being in the state.

The limit probabilities for a finite Markov chain satisfy the following system of linear equations[11]:

$$\pi_j = \sum_{i=1}^{M} \pi_i P_{ij}, \qquad for\ j = 1, 2, ..., M,\ subject\ to\ \sum_{j=1}^{M} \pi_j = 1,$$

which can be used to determine π_j $(1 \leq j \leq M)$. All limit probabilities $\{\pi_j \mid j = 1, 2, ...M\}$ comprise a so-called long-run distribution for the Markov chain.

Since we consider, in this section, the situation in which Q is a large (cost) query and the contention states in the dynamic environment change frequently, it is expected that there are many transitions during the execution of Q. Since π_i represents the long-run portion of time for the Markov chain being in S_i, $\pi_i * C(Q)$ is the amount of cost incurred in state S_i. Hence $(\pi_i * C(Q))/C(Q, S_i)$

is the portion of work done for Q in S_i. Clearly, the portions of work done for Q in all states should add to 1, i.e., $\sum_{i=1}^{M} \frac{\pi_i * C(Q)}{C(Q,S_i)} = 1$. Solving this equation, we have

$$C(Q) = 1/[\sum_{i=1}^{M} \frac{\pi_i}{C(Q,S_i)}]. \tag{7}$$

Since $C(Q,S_i)$ can be estimated by using the dynamic cost models with qualitative variables, discussed in Section 2, formula (7) can be used to estimate the cost of query Q in the dynamic environment in which the contention states change rapidly.

Comparing the above probabilistic approach with the single state analysis approach, we notice that the cost estimate of a query given by the single state analysis approach is identical to the one given by the probabilistic approach when the limit probability for the selected single state is 1 (i.e., the limit probabilities for other states are 0). However, in general, more than one state has a non-zero limit probability. The single state analysis is, therefore, not an appropriate approach for such a rapidly changing environment.

Comparing the probabilistic approach with the average cost analysis approach, we have the following propositions:

Proposition 3. *If* $\pi_1/C(Q,S_1) = \pi_2/C(Q,S_2) = ... = \pi_M/C(Q,S_M)$, *then* $C(Q) = \bar{C}(Q)$, *where* $C(Q)$ *and* $\bar{C}(Q)$ *are the Markov cost estimate from (7) and the average cost estimate, respectively.*

Proof. Since $\pi_i/C(Q,S_i) = \pi_j/C(Q,S_j)$, we have $\pi_i = \pi_j * C(Q,S_i)/C(Q,S_j)$. From (7) and the average cost estimate formula, we have

$$\frac{\bar{C}(Q)}{C(Q)} = \frac{1}{M}\sum_{i=1}^{M} C(Q,S_i) \sum_{j=1}^{M} \pi_j/C(Q,S_j) = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{M} C(Q,S_i) * \pi_j/C(Q,S_j)$$

$$= \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{M} \pi_i = \frac{1}{M}\sum_{i=1}^{M} M * \pi_i = \sum_{i=1}^{M} \pi_i = 1$$

Therefore, $C(Q) = \bar{C}(Q)$. □

Although $\pi_i/C(Q,S_i)$ does not have any physical meaning, $\pi_i * C(Q)/C(Q,S_i)$ is the portion of work done for Q in state S_i. Therefore, the assumption in the proposition implies that each state completes an equal portion of work for Q. In such a case, the cost estimates from the probabilistic approach and average cost analysis are the same. However, such an assumption is usually not met. Since the probabilistic approach can cope with different limit probability distributions, it is expected to give better cost estimates.

Proposition 4. *If* $\pi_1 = \pi_2 = ... = \pi_M = 1/M$, *then* $C(Q) < \bar{C}(Q)$.

Proof. Similar to the proof for Proposition 2, omitted. □

Proposition 4 states that the cost estimate for a query by the average cost analysis is larger than the cost estimate for the query by the probabilistic approach when the limit probabilities for all states are the same. This phenomenon can also be explained by the different working rates in the contention states, like the explanation given for Proposition 2.

5 Experimental Results

To validate the cost estimation techniques proposed in the previous sections, experiments were conducted using a multidatabase system prototype, named *CORDS-MDBS*[2]. Two commercial DBMSs, i.e., Oracle 8.0 and DB2 5.0, were used as component database systems running under Solaris 5.1 on two SUN UltraSparc 2 workstations. Fig. 3 shows the experimental environment. To test the techniques in various dynamic environments, we developed a load builder which can generate dynamic system loads to simulate various dynamic environments following different load curves (for the fractional analysis approach) or retention probability distributions (for the probabilistic approach) for contention states.

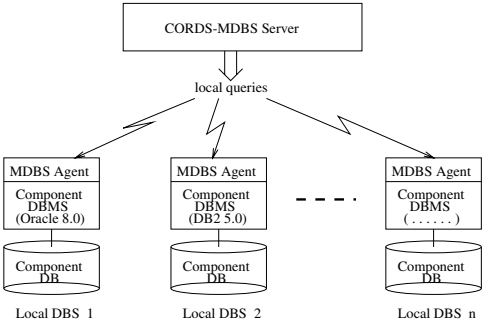


Fig. 3. Experimental Environment

The table schemas in the component databases used in the experiments were the same as those[4] in [15, 17]. More specifically, each component database contains 12 tables $R_i(a_1, a_2, ..., a_n)$ $(1 \leq i \leq 12; 1 \leq n \leq 13)$ with all integer columns. The data in the tables are randomly generated using different ranges for different columns to achieve various selectivities. The table cardinalities range from 3,000 to 250,000. Each table has some indexed columns. For more details of the test database, please refer to [15, 17].

Since the costs of unary queries are usually not large and the techniques in this paper are for large cost queries, we chose a join query class for our experiments. Following the multi-states query sampling method in Section 2, we drew a sample of queries from the query class and executed them in a dynamic

[4] The size of each table is ten times larger than that in [15, 17].

multidatabase environment. Based on the observed costs of sample queries, our cost model building tool automatically selects significant variables and uses them together with a qualitative variable (represented by a set of indicator variables) indicating system contention states to develop a cost model for the query class. Our tool also applies some statistical measures to validate the significance of the cost model. Table 1 shows the cost models developed for the query class

Table 1. Cost Models for a Join Query Class in a Dynamic MDBS Environment (RN_J — result table size; TN_{J1}, TN_{J2} — 1st and 2nd intermediate table sizes; TN_{J12} — $TN_{J1} * TN_{J2}$; RL_J — result table tuple length; L_{J1}, L_{J2} — 1st and 2nd operand table tuple lengths; TZ_{J1} — $N_{J1} * L_{J1}$; TZ_{J2} — $N_{J2} * L_{J2}$; N_{J1}, N_{J2} — 1st and 2nd operand table sizes.)

	(Dynamic) Cost Model with Qualitative Variable			
Component DBMS 1 (Oracle)	$(0.7419e{+}1 - 0.1169e{+}3 * Z_4 + 0.2748e{+}2 * Z_3 - 0.3963e{+}2 * Z_2$ $+0.3626e{+}2 * Z_1) + (0.1131e{-}2 + 0.2212e{-}2 * Z_4 + 0.4383e{-}2 * Z_3$ $+0.5517e{-}2 * Z_2 + 0.6588e{-}2 * Z_1) * RN_J + (0.4952e{-}3 - 0.5211e{-}3 * Z_4$ $-0.2522e{-}3 * Z_3 + 0.3308e{-}3 * Z_2 + 0.2617e{-}2 * Z_1) * TN_{J1} +$ $(-0.4691e{-}3 + 0.6974e{-}3 * Z_4 + 0.9641e{-}3 * Z_3 + 0.1900e{-}2 * Z_2$ $+0.3194e{-}2 * Z_1) * TN_{J2} + (-0.1121e{+}1 + 0.5489e{+}1 * Z_4$ $+0.1419e{+}2 * Z_3 + 0.7953e{+}1 * Z_2 + 0.1452e{+}2 * Z_1) * RL_J +$ $(-0.1995e{+}2 + 0.3754e{+}2 * Z_4 + 0.2510e{+}2 * Z_3 + 0.3139e{+}2 * Z_2$ $+0.8524e{+}1 * Z_1) * L_{J1} + (0.1647e{+}2 - 0.2275e{+}2 * Z_4$ $-0.3715e{+}2 * Z_3 - 0.2881e{+}2 * Z_2 - 0.2503e{+}2 * Z_1) * L_{J2}$			
	coef. of multi. determination	std. error of estimation	avg. sample cost (sec.)	F-statistics (critical value at $\alpha = 0.01$)
	0.9992	0.1708e+3	0.1178e+4	16166.13 (> 1.25)
Component DBMS 2 (DB2)	(Dynamic) Cost Model with Qualitative Variable			
	$(-0.1529e{+}2 - 0.1923e{+}1 * Z_3 + 0.1178e{+}2 * Z_2 + 0.3510e{+}1 * Z_1)$ $+ (0.4171e{-}7 - 0.1697e{-}7 * Z_3 + 0.6500e{-}7 * Z_2 + 0.9597e{-}6 * Z_1) * TN_{J12}$ $+ (0.8934e{-}3 + 0.1697e{-}2 * Z_3 + 0.2478e{-}2 * Z_2 + 0.3554e{-}2 * Z_1) * RN_J$ $+ (0.2763e{-}3 - 0.1106e{-}2 * Z_3 - 0.4020e{-}3 * Z_2 + 0.7719e{-}2 * Z_1) * TN_{J1}$ $+ (0.1234e{-}4 + 0.4639e{-}4 * Z_3 + 0.2241e{-}4 * Z_2 + 0.3693e{-}3 * Z_1) * TZ_{J1}$ $+ (0.3684e{-}4 + 0.6292e{-}4 * Z_3 + 0.1345e{-}4 * Z_2 - 0.1768e{-}3 * Z_1) * TZ_{J2}$			
	coef. of multi. determination	std. error of estimation	avg. sample cost (sec.)	F-statistics (critical value at $\alpha = 0.01$)
	0.9963	0.2436e+3	0.7860e+3	5108.04 (> 1.41)

on the two component DBMSs, i.e., Oracle and DB2. The coefficient of total determination indicates that both cost models can capture over 99% variations in the query costs. The F-test also shows that both cost models are useful. Between the two models, the one for Oracle is even better.

To further validate the cost models, we ran some randomly-generated test queries[5] in the dynamic environment under the restriction that each query only experiences a (random) single contention state. We applied the dynamic cost

[5] The test queries used in this paper are the same as those in [15].

models in Table 1 to estimate the costs of the test queries and compared the estimated costs with their observed costs as well as the estimated costs using a static cost model[6]. The comparison results are shown in Fig.'s 4 and 5. From the figures we can see that the estimated costs given by the dynamic cost model are much better than the ones given by the static cost model.

However, assuming a query to experience one state may be only valid for small (cost) queries. For large (cost) queries, the execution of a query may experience more than one contention state. The techniques presented in Sections 3 and 4 should be applied to estimate the cost of such a query.

In the experiments to evaluate the effectiveness of the fractional analysis technique, the "shape" of load curve in Fig. 1 is assumed. However, for simplicity, (1) the "noon" contention state S_1 is split into two state occurrences, e.g., the contention states sequence $S_4, S_3, S_2, S_1, S_1, S_2, S_3, S_4$ was used for experiments on DB2; and (2) the load curve repeats its pattern (the contention states sequence) periodically so that the queries that cannot be finished within the current cycle can be completed within the following cycle(s). The initial starting state for each test query was randomly selected from the contention states sequence. Different changing patterns ("increase", "decrease", "equal", and "random") of time durations for the contention state occurrences along the curve were tested in our experiments. Fig.'s 6 and 7 show the experimental results for the "random" case, which well represents all other cases. The following observations can be obtained from our experiments:

- The fractional analysis technique can give good cost estimates for the test queries in a gradually and smoothly changing environment. Most cost estimates in the experiments have relative errors within 30%.
- The cost estimates given by the fractional analysis are clearly better than the ones given by the average cost analysis or the initial single state analysis for most cases.
- There are some cases (i.e., queries completed entirely within one state) in which the cost estimates from the fractional analysis and the initial single state analysis are the same, which is consistent with our theoretical analysis.
- The average costs are larger than the estimated costs given by the fractional analysis when a query spent an equal (accumulated) amount of time in every state, which is consistent with Proposition 2 we showed in Section 3.
- The accuracy of the cost model with a qualitative variable used to estimate $C(Q, S_i)$ of query Q for state S_i is important to the fractional analysis. Since we obtained a better cost model for Oracle, its fractional analysis results are overall more accurate than the ones for DB2.

In the experiments to evaluate the effectiveness of the probabilistic approach, we tested various dynamic environments with different retention probability changing patterns ("increase", "decrease", "equal", and "random"). Since an environment with an equal retention probability for all contention states may not yield an equal limit probability for every contention state, we also tested

[6] The static cost model was developed by using our static sampling query method in [15], i.e., assuming that the environment has only one state (static).

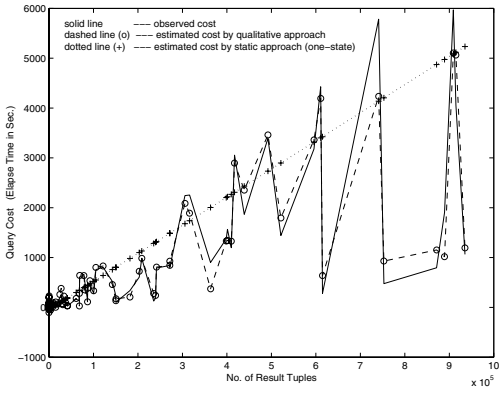


Fig. 4. Estimated Costs for Test Queries via Qualitative Approach on Oracle 8.0

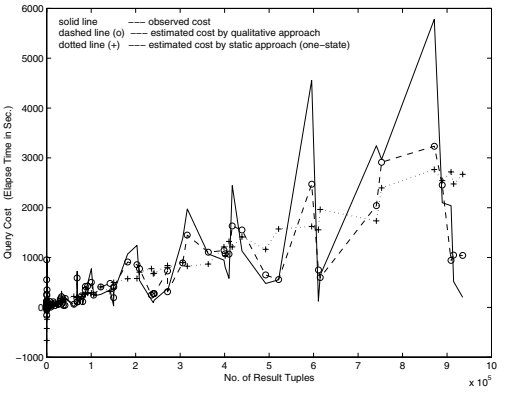


Fig. 5. Estimated Costs for Test Queries via Qualitative Approach on DB2 5.0

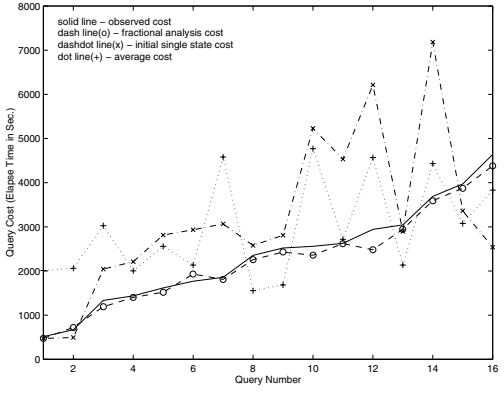


Fig. 6. Estimated Costs for Test Queries via Fractional Analysis on Oracle 8.0

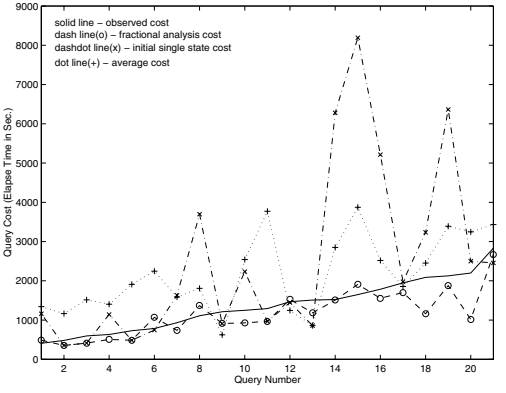


Fig. 7. Estimated Costs for Test Queries via Fractional Analysis on DB2 5.0

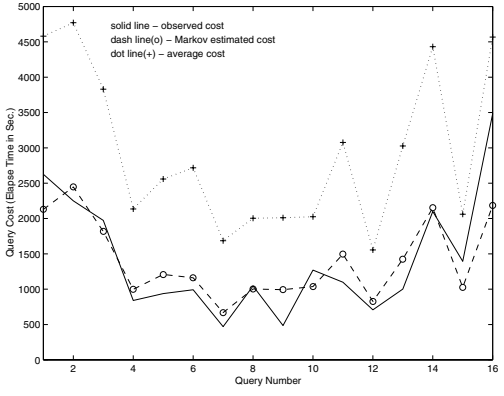


Fig. 8. Estimated Costs for Test Queries via Probabilistic Approach on Oracle 8.0

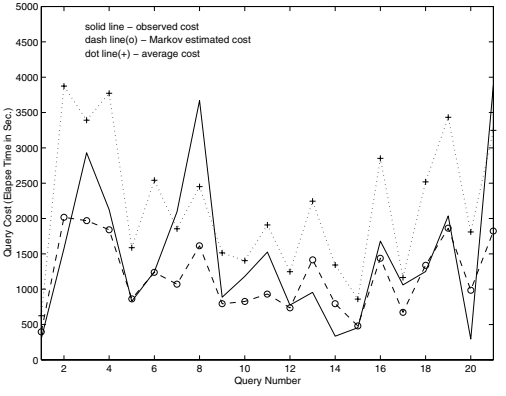


Fig. 9. Estimated Costs for Test Queries via Probabilistic Approach on DB2 5.0

the "equal" limit probability case. The experimental results for the "random" case, which well represents other cases, are shown in Fig.'s 8 and 9. From the experiments, we can obtain similar observations that we had (see above) for the fractional analysis (except that the single state analysis was not considered).

In summary, our experimental results demonstrate that the qualitative approach, fractional analysis approach and probabilistic approach comprise a promising suite of techniques in estimating local query costs for a dynamic multidatabase environment.

6 Conclusion

A major challenge for performing global query optimization in an MDBS is that some local cost information may be unavailable at the global level. Most techniques suggested so far in the literature considered only static system environments, namely, assuming the environment never changes. However, the cost of a query changes dramatically in a realistic dynamic environment.

To solve the problem, we have studied several new techniques. In our recent work[18], we employed a multi-states query sampling method to develop cost models with qualitative variables indicating the system contention states. For queries experiencing a single state, such developed cost models can be used to directly estimate their costs. To estimate the costs of (large) queries experiencing multiple states, we propose two new techniques, i.e., fractional analysis and probabilistic approach, to estimate their costs in this paper. The first one is suitable for a gradually and smoothly changing environment, while the latter is suitable for a rapidly and randomly changing environment. The cost formulas in both cases are derived. Their properties are analyzed. Note that although the two new techniques make use of the cost models developed by the multi-states query sampling method, it is not required; in other words, any method that can estimate the cost of a query in each contention state can be used together with the two techniques proposed in this paper. To validate the effectiveness of the new techniques, we conducted extensive experiments. Our experimental results demonstrate that the presented techniques are quite promising in estimating query costs in a dynamic multidatabase environment. Their produced cost estimates for most queries have relative errors within 30%.

Dynamic environmental factors were ignored in existing cost models for database systems due to lack of appropriate techniques. The work reported in this paper has shown some promise in solving the problem. However, our work is just the beginning of work needed to be done in order to completely solve all related issues.

Acknowledgements

We would like to thank Thanikachalam Ponnusamy for some study on the dynamic environments for real-world application systems. We are grateful to Amira Rahal-Arabi for her help in running some experiments. We would also like to

thank Per-Åke (Paul) Larson and Frank Olken for their valuable suggestions and comments for the work reported in this paper.

References

1. S. Adali, K. S. Candan, Y. Papakonstantinou, and V. S. Subrahmanian: Query caching and optimization in distributed mediator systems. In *Proc. of ACM SIGMOD Inf. Conf. on Management of Data*, pp 137–48 (1996)
2. G. Attaluri, D. Bradshaw, N. Coburn, P.-A. Larson, P. Martin, A. Silberschatz, J. Slonim, and Q. Zhu: The CORDS multidatabase project. *IBM Systems Journal*, **34** (1995) 39–62
3. W. Du, R. Krishnamurthy, and M. C. Shan: Query optimization in heterogeneous DBMS. In *Proceedings of the 18th VLDB Conference*, pp 277–91 (1992)
4. W. Du, M. C. Shan, and U. Dayal: Reducing multidatabase query response time by tree balancin. In *Proc. of SIGMOD'95*, pp 293 – 303 (1995)
5. C. Evrendilek, A. Dogac, S. Nural, and F. Ozcan: Multidatabase query optimization. *Distributed and Parallel Databases*, **5** (1997) 77–114
6. G. Gardarin, F. Sha, and Z.-H. Tang: Calibrating the query optimizer cost model of IRO-DB, an object-oriented federated database system. In *Proceedings of the 22nd VLDB Conference*, pp 378–89 (1996)
7. C. Lee and C.-J. Chen: Query optimization in multidatabase systems considering schema conflicts. *IEEE Trans. on Knowledge and Data Eng.*, **9** (1997) 941–55
8. W. Litwin, L. Mark, and N. Roussopoulos: Interoperability of multiple autonomous databases. *ACM Computing Surveys*, **22** (1990) 267–293
9. H. Lu, B.-C. Ooi, and C.-H. Goh: On global multidatabase query optimization. *SIGMOD Record*, **21** (1992) 6–11
10. H. Naacke, *et al.*: Leveraging mediator cost models with heterogeneous data sources. In *Proc. of 14th Int'l Conf. on Data Eng.*, pp 351–60 (1998)
11. E. Parzen: *Stochastic Processes*. Holden-Day, Inc. (1962)
12. M. T. Roth, F. Ozcan, and L. M. Haas: Cost models DO matter: providing cost information for diverse data sources in a federated system. In *Proc. of the 25th VLDB Conf.*, pp 599 – 610 (1999)
13. A. P. Sheth and J. A. Larson: Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, **22** (1990) 183–236
14. Qiang Zhu and P.-A. Larson: A fuzzy query optimization approach for multidatabase systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **5** (1997) 701 – 22
15. Qiang Zhu and P.-A. Larson: Solving local cost estimation problem for global query optimization in multidatabase systems. *Distributed and Parallel Databases*, **6** (1998) 373–420
16. Qiang Zhu and P.-A. Larson: Building regression cost models for multidatabase systems. In *Proceedings of the 4th IEEE International Conference on Parallel and Distributed Information Systems*, pp 220–31 (1996)
17. Qiang Zhu and P.-A. Larson: A query sampling method for estimating local cost parameters in a multidatabase system. In *Proceedings of the 10th IEEE International Conference on Data Engineering*, pp 144–53 (1994)
18. Qiang Zhu, Y. Sun, and S. Motheramgari: Developing cost models with qualitative variables for dynamic multidatabase environments. In *Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE'2000)*, pp 413-424 (2000)

Lazy Query Enrichment: A Method for Indexing Large Specialized Document Bases with Morphology and Concept Hierarchy

Alexander F. Gelbukh

Center for Computing Research (CIC),
National Polytechnic Institute (IPN),
Av. Juan Dios Bátiz s/n esq. Mendizábal,
Col. Zacatenco, C.P. 07738, D.F., Mexico
gelbukh@cic.ipn.mx

Abstract. A full-text information retrieval system has to deal with various phenomena of string equivalence: ignore case matching, morphological inflection, derivation, synonymy, and hyponymy or hyperonymy. Technically, this can be handled either at the time of indexing by reducing equivalent strings to a common form or at the time of query processing by enriching the query with the whole set of the equivalent forms. We argue for that the latter way allows for greater flexibility and easier maintenance, while being more affordable than it is usually considered. Our proposal consists in enriching the query only with those forms that really appear in the document base. Our experiments with a thesaurus-based information retrieval system showed only insignificant increase of the query size on average with a 200-megabyte document base, even with highly inflective Spanish language.

1 Introduction*

Any information retrieval system has to somehow deal with the problem of non-literal matching of the query and the document keywords. For example, given a query *computer*, the system should be able to retrieve (or not, depending on the user-defined settings and query options) the documents containing the strings *Computer*, *computers*, *computation*, *mainframe*, *motherboard*, *Internet*, etc. There are two places in the system architecture where this problem can be dealt with:

- At the moment of indexing the documents – *index enrichment* – or
- At the moment of processing of the specific query – *query enrichment*.

The former technique is most commonly used due to apparently prohibitively serious problems caused by the latter one. We will show, however, that the former method has its own disadvantages, and that the problems of the latter method can be efficiently solved.

* Work done under partial support of the Senate of Mexican Republic, CONACyT (Mexico), and CGEPI-IPN (Mexico).

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 526-535, 2000.

Our main motivation was the development of an information retrieval system for the Senate of Mexican Republic. Our customer's priorities were the following:

- The *quality* of search; expressive power, and flexibility of the query language.
- Small *index size* and low load on the server during *updating* the database contents.
- *Minimal changes* to the existing maintenance technology and database structure.
- The system was to be *operational* while the dictionaries and grammars are under development, and the improvements and corrections to them were to be immediately available to the users.

On the other hand, computational efficiency of processing of a single query was of lower priority since the number of the users was rather limited. The characteristic properties of the document base at hand were the following:

- Large size, in the order of a gigabyte, to be extended to several gigabytes,
- Specialized contents with limited variety of lexicon and syntactic constructions,
- Still, unrestricted language with the possibility of occasional use of any word form.

In this work, we are interested in a flexible, computationally efficient, conceptually simple, and easily maintainable solution of the problem of non-literal matching under the requirements and circumstances listed above.

1.1 Related Work

There is a vast literature on approximate string matching; a variety of data structures, such as tries, B-trees, etc. were suggested [1, 5, 8]. These works are based on implicit or explicit letter patterns. However, here we consider the problem of matching arbitrary word sets that might not share any simple letter pattern. E.g., the strings *dormía*, *duermo*, and *durmiendo* are forms of the same Spanish verb *dormir* 'to sleep;' the strings *church*, *priest*, and *pilgrim* represent the same concept *religion* though they do not match any particular letter pattern for approximate string matching to be applied.

The problem of generating and matching the word forms in various languages, including English and Spanish, is well studied in linguistics. Various methods and data structures are suggested in computational linguistics for handling the corresponding dictionaries and morpheme lists [3, 9, 10]. However, in this article we do not discuss the problems of natural language morphology. Instead, we are interested in application of a morphological analyzer to the purely database management task of retrieval of a keyword in all its forms. The list of the word forms is supposed to be already known, while these forms are not supposed to match any particular letter pattern.

The use of concept hierarchies for topical document analysis was suggested in [6] and applied to the information retrieval tasks in [4]. Various large hierarchical thesauri have been compiled [2, 7, 12]. Again, here we are interested not in the compilation or tuning of the thesaurus itself but rather in the way the documents relevant for a specific node can be in practice found in an existing large document base.

2 Types of Non-literal String Matching

Here we will discuss in more detail the types of the strings that the user might want, or not want, to be matched. An important point in each case is the great degree of flexibility necessary to meet the requirements of a specific user or a specific search.

2.1 Letter Case

This is the simplest type of non-literal matching: usually the strings like *computer*, *Computer*, *COMPUTER*, and *ComPuTer* are to be considered equivalent. On the other case, the user should have an option to turn off such equivalence, e.g., to find *Bill* or *Mainframe* (personal names) but not *bill* nor *mainframe*, *CIS* (the name of an organization) but not *Cis* (personal name).

2.2 Morphology

The second class of strings that frequently are considered equivalent are the word forms of the same lexeme: *compute*, *computing*, *computed*, or its derivational variants: *computer*, *computation*, *computational*, *computability*.

On the other hand, matching the morphological forms of the same stem is not always desirable. For example, a user can be interested in *computers*, but not in *computation*. Very annoying is morphological reduction of ambiguous forms, especially in highly inflective languages such as Spanish. E.g., the Spanish verb *comer* 'to eat' form about 700 morphological variants like *comiste* 'you ate' or *comiéndotela* 'you eating it up', one of which – namely *como* 'I eat' – happens to be homonymous with a very frequently used conjunction *como* 'as,' 'how.' Thus, to find the documents with the Spanish lexeme *comer* with a reasonable precision, one has to sacrifice recall a little bit by forming the query as "all word forms of *comer* but *como*."[1]

Thus, the user should be able to control the application and the degree of morphological normalization applied to the query by the system.

2.3 Concept Hierarchy

The third class of the words that might be considered equivalent are synonyms (*processor*/*CPU*), hyponyms (*computer*/*mainframe*), hypernyms (*computer*/*device*), and possibly other related words.

A dictionary hierarchical thesaurus is used to provide this option to the user. In our system, we use a 33,000-word dictionary organized in a deep hierarchy of related concepts, similar to, and in part derived from, the Roget thesaurus [2]. By related

[1] Note that this effect cannot be achieved with a simple logical expression "all documents containing the word forms of *comer* but those containing the word *como*" since its meaning is not equivalent to the desired one. Namely, the recall with such a query would be extremely low since nearly any Spanish document does contain the string *como* 'as,' 'how.'

concepts, we mean not only the *is-a* relationship, but also other words that are of interest to the user looking for the documents on a given topic [6]. For example, the entry for *religion* contains such words as *Bible*, *priest*, *pray*, *church*, *pilgrim*, etc. Thus, the user looking for the documents on *religion* will be offered a document that mentions *Bible*. Optionally the degree of such relationship can be weighted quantitatively to measure the relevance of the found document [4].

Obviously, the user should have a full control over the set of words to be considered equivalent to the query keyword(s), since the words that are synonymous for one user might well be very different for another. The following options, or some their combination, are of particular interest:

1. *All instances* of a given concept, i.e., all words below a given node.
2. *Near-immediate instances* of a given concept, i.e., the words below a given node but not deeper than *n* levels.
3. *Similar* concepts, the words located in the concept tree not farther than *n* steps from the given one, be the steps in the down, up, or horizontal direction in the tree.
4. *General* concepts, i.e., the words of which the given node is an instance.

3 Index Enrichment

In the previous section, we discussed four cases of identity of the strings: letter case, morphologically inflected forms, synonyms, and a concept tree. A naïve – and the most frequently used – approach to represent the first three cases of identity is *index reduction*: at the moment of indexing, all letters are reduced, say, to lowercase; all word forms are reduced to the main form (*computing*, *computed*, *computes, computation, computer* → *compute*), and synonyms are replaced with one chosen representative (*CPU* → *processor*). The latter case – a concept tree – can be handled by additionally indexing each document with the hypernyms of the words it contains (*mainframe* → *computer*, *device*, *artifact*); with this method, a query "*devices*" will retrieve also *mainframe*.

In this article, we argue that this naïve approach has serious disadvantages. First of all, as we have shown in the previous section, depending on the desired precision/recall ratio, the user might *not* want such strings to be considered identical. Thus, indexing process should not cause any loss of information – i.e., all the letter strings should appear in the index *as is*, without any change, even in the letter case (i.e., reducing to lowercase). To achieve this, the strings reduced in letter case, or morphologically, or by a thesaurus should appear in the index in addition to (rather than instead of) the original strings. We call this process index enrichment.

To allow for certain flexibility of the queries, some improvements to this indexing scheme can be suggested. For instance, the additional keys are to be marked somehow to be distinguishable, if needed, from the original ones. Other possible improvements will be discussed in section 0. However, the index enrichment method presents some inherent problems:

- *Lack of flexibility*. Only the types of queries for which the index was specifically designed can be executed. The user cannot choose what words of a given set are to b considered equivalent, e.g., "all word forms of *compute* but *computing*;" see also the discussion of the example with the Spanish *comer* in section 2.2, and also the footnote there.

- *Lager index*. Unlike index reduction, index enrichment can significantly – from twice to tenfold, depending of the use of only morphology or also a thesaurus – increase the index size.
- *Maintenance difficulties*. Too close coupling of the indexing process and complex lingware – the morphological analyzer and the thesaurus – presents both organizational and technical problems.
  - Adding the intelligent search engine to an existing database maintenance technology requires significant changes in the latter, which implies changing existing software and documentation, training the maintenance engineers, etc.
  - Unlike stable database maintenance procedures, complex dictionary-based lingware tends to be – at least initially – in constant development. With index reduction or enrichment, each time a change is made to the lingware, the whole database is to be re-indexed, which is often not affordable, especially when the linguistic processing is slow and resource-consuming. On the other hand, delaying re-indexing for a long time greatly discourages any improvements to the lingware, from the point of view of both the developers and the customer.

4 Query Enrichment

An alternative to handling non-literal string matching at the stage of indexing is handling it at the stage of query processing. A naïve approach to this method is the following. The letter strings found in the documents are indexed *as is*, without any changes. Then, at the moment of query processing, the user query is automatically substituted with an appropriate logical expression, e.g., the query "*compute* and *matrix*" internally is executed as "(*compute* OR *computes* OR *computed* OR *computing*) and (*matrix* OR *matrixes* OR *matrices*)." We call this procedure query enrichment; it is also known as query expansion [11].

This method does not present any of the problems listed in the previous section. Namely, it has the following advantages over index enrichment or reduction:

- *Flexibility*. The user can edit the resulting expression to achieve any desired combination. For example, the query "all forms of the Spanish verb *comer* but *como*" can be naturally expressed by the user and processed by the system.
- *Easier scoring*. Enrichment can be done gradually: first the unmodified query is performed; only if it does not give the necessary results, the query words are substituted with their most close equivalents and the search is repeated; only if this search does not give the necessary results, the query words are substituted with their less close equivalents, etc. With this, the documents that closer corresponds to the query will be found first.
- *Smaller index* as compared with index enrichment. Only the strings literally present in the document are present in the index.
- *Easy maintenance*. The indexing procedure is trivial and does not include, nor depends on, any lingware. No changes to the existing non-intelligent indexing technology are necessary when adding an intelligent search engine to an operational database. No re-indexing is necessary when changes are made to the lingware, and such changes are available immediately to the user.

However, the disadvantages of this naïve approach are so obvious that it cannot be considered as a practical alternative. Namely, the following two problems render such a method practically unusable:

- *Too large queries*. As we have mentioned, the Spanish verb *comer* form about 700 variants, which results in too large query. With a thesaurus, the concept *Europe* would contribute to the query all countries, cities, rivers, mountains, nations, types of food, etc. specific for Europe. In addition, each of these strings should be capitalized in all possible ways.
- *Generation*. It is well known that generating all forms of a given lexeme (*compute* → *compute*, *computing*, ..., *uncomputable*, ...) is a task significantly more difficult than guessing the correct main form or stem of a given word form (*compute*, *computing*, *uncomputable* → *compute*). In case of a heuristic-based (rather than dictionary-based) morphological algorithm, the number of hypotheses in form generation is usually much greater than in reducing to the stem.

  In the next section, we will show how these problems can be worked around.

5 Lazy Query Enrichment

The improvement we suggest for the method of query enrichment consists in including into the enriched query only the strings known to be present in at least one document of the given database. Since the diversity of language in a specialized document base (such as legal, medicine, etc. texts) is relatively low, only a small fraction of all possible forms of a word or sub-concepts of a concept are present in the database, which greatly reduces the size of the enriched query. At the same time, when applied to the specific database, such a reduced query is equivalent to the fully enriched query. We call this modification of the enrichment procedure *lazy enrichment*.

The process of lazy query enrichment can be sketched as following.

- list A of all strings that appear at least once in the given database is build.
- This relatively small list is indexed as described in section 0, which produces an index table such as the following:

Letter case	
String	Identifier
computer	computer
Computer	computer
CompuTer	computer

Derivation	
String	Identifier
computes	compute
computing	compute
uncomputability	compute

Thesaurus	
String	Identifier
mainframe	computer
mainframe	device
mainframe	artifact

  Here by the identifier (ID) we mean a reduced form, such as reduced to the lowercase, morphologically reduced to the main form, promoted up the tree in the thesaurus, etc., see section 0 (we do not show in this table the improvements discussed in the sections 0 and 0).
- At the stage of processing the query, each keyword of the query is subject to appropriate reduction depending on the user-defined option, for example, to morphological reduction to its main form, e.g., *computable* → *compute*, thus giving a potential ID. In case of ambiguity, all potential IDs are obtained.

- The ID(s) for each query keyword are looked up in the right column of the table, and the word is substituted with the list of the corresponding literal strings found in the left column.

For example, with the table above, the query "what things are *computable*?" is transformed first into the query "ID = compute" and after the lookup in the table, into the query "*computes* OR *computing* OR *uncomputability*." Note that it does not contain such strings as *compute*, *computed*, nor the very source form *computable* if they do not occur in the documents in the database.

To process a complex query, such as, for example, a query of the type 3 discussed in the section 2.3, the thesaurus is navigated correspondingly and the query is first built as a disjunction of the relevant lexemes (concepts) and then is enriched through the index table as described above.

The suggested modification of the of the query enrichment method does not have any disadvantages of the latter, thus presenting the following advantages as compared with full query enrichment:

- *Smaller queries*. Only the words really appearing in the database are included into the query. The difference is especially sensible in the languages with developed morphology. For example, of about 700 forms of the frequently used Spanish verb *comer* 'to eat,' in the database of the Senate of Mexican Republic appeared only 29; of more than 100 forms of *falsificar* 'to falsify,' appeared only 11, etc.
- *Only reduction*. The algorithm does not use any generation; instead, only reduction is used, such as reduction to lowercase or morphological reduction. This greatly simplifies the lingware, allowing for a rather simple heuristic-based morphological algorithm.

Of course, the suggested method still has some disadvantages as compared with the full query enrichment or index enrichment methods, among them the following ones:

- *Need to maintain the list of strings*. As compared to the full query enrichment, the suggested method requires to maintain an additional data structure. In the next section we will show that this does not present serious maintenance problems.
- *Still increase in query size*. As compared to the index enrichment, the queries are still increased in size, though not as much as with full query enrichment.

6 System Architecture

Our system is built on top of the existing operational technology treated as a black box. The user query is intercepted, analyzed, and substituted with an enriched query using the lazy query enrichment technique. The new query is presented to the user in a user-friendly form for possible editing. At this stage, gradual enrichment described in the section 0 can be applied.

In our case, the system is based on Informix DataBlade platform. This DBMS allows for specifying with each query a so-called synonym list. It looks much like a synonym dictionary: for each headword, it lists the strings that will be treated by the system as equivalents for each occurrence of the headword in the query. This feature proved to be ideal for the query enrichment technique: instead of enriching the query "*computers* AND *linguistics*" as "(*computer* OR *computers* OR *computational* OR ...)

AND (*linguistics* OR *language* OR *linguist* OR ...)" we leave the query untouched and provide as a search parameter a synonym list containing the entries such as:

computer: *computers*, *computational*, *compute*, *PC*, *workstation*, *mainframe*, ...
linguistics: *language*, *linguist*, *linguistic*, *English*, *Spanish*, *HPSG*, ...

To maintain the list of strings that occur in the database, the database index can be used directly to update the list each time the database contents updated. Alternatively, if such an internal structure of the existing DBMS is not easily accessible, the documents can be indexed by a separate program.

In our system, for the index updating procedure to be independent from the pre-existing database updating technology, an agent periodically searches the database for the new documents, processes them, and marks as processed. After the list of strings has been updated, the new strings are subject to all necessary types of reduction (morphological forms, synonyms, etc.) as described in Section 0. Finally, if the lingware or dictionaries are updated, the whole list is re-indexed; due to small size of the list, this does not present any serious problem.

7 Experimental Results

We investigated a 200 MB subset of the database of the Mexican Senate containing a representative mixture of the discourses of the Senators, laws, and other working documents of the Senate. The corpus contained 21,378,740 letter strings (running words), of them, only 174,386 strings different ones (0.8%). Then we reduced the strings in various ways. Obviously, the ratio of such reduction is equal to the ratio of inflating the query when lazy query enrichment is used.

First, lowercase reduction gave 102,715 different strings, which shows that with only letter case equivalences taken into account, the query is inflated insignificantly – less than twice.

Morphological reduction using a simple postfix removing heuristics showed that the database used 55,489 different stems. Therefore, the average number of strings per stem – i.e., the average ratio of lazy query enrichment using only morphology – was about 4. Here we fully distinguished lowercase and uppercase letters in query enrichment: e.g., the stem *cultiv-* was represented by three strings: *Cultiva*, *cultiva*, and *cultivaron*.

Thesaurus-based enrichment showed a bit less promising results. Here are two examples. In our dictionary, the concept *a Mexican city* consists of 2,413 names. The database happened to mention 1,130 such words with case ignoring comparison, or 1,780 strings if uppercase and lowercase letters are distinguished. The concept *body parts* in our dictionary is represented by 97 words; the database happened to mention 55 of them, or 86 strings with letter case distinguished.

Thus, with thesaurus-based enrichment, the query inflation ratio is high and might be not affordable in a practical system.[2] However, as we have mentioned, due to the

[2] In our system, the application of query enrichment is based on the Informix DataBlade's synonym list feature, as described in the section 0. Our tests showed that this feature works with the entry size up to 3,000 synonyms for one headword. Thus, with this particular platform, thousand-fold enrichment is still possible.

very nature of a thesaurus reflecting "general" knowledge that often proves to be not suitable for a particular user, as well as due to rather low quality of existing dictionaries, this type of enrichment especially needs in the degree of user's control that cannot be provided by index enrichment. Therefore, we consider the disadvantage of the query enrichment method to be technical, i.e., temporal, while its advantage – a greater degree of control – to be fundamental. Note that as the dictionary is being elaborated, the inflation ratio will not significantly increase since the new words being added to the dictionary are of low frequency in the texts. In the next section, a possible workaround for the problem of too high query inflation will be discussed.

8 Future Work: A Combined Technique

Even with the proposed technique, query enrichment still slows down the system by inflating the user query, especially in case of thesaurus-based enrichment. On the other hand, index enrichment has an advantage of using the context of the word:

- Multiword expressions and idioms in the thesaurus, such as *hot dog*, can be handled naturally at the stage of indexing of the full text of the document.
- The words can be disambiguated in context; e.g., with the query "*wells*," the text *oil well* will be found while *he did it well* not.
- The structure of the document can be taken into account, e.g., words in the title or abstract can be indexed differently from the words in the main text.
- The global properties of the document not related with any specific word in it, such as the main topic of the document [4, 6], can be used for indexing.

To provide these features without sacrificing the flexibility of the query language, the index enrichment can be used in combination with the query enrichment. The first step to such combination is the following. Both methods are implemented in the system; in particular, the documents are indexed with index enrichment as explained in section 0. The user queries of standard types such as full morphological reduction or a full thesaurus-based query (see section 2.3), are processed fast with the enriched index without any query enrichment. In the rather rare case when the user somehow modifies the list of strings to be matched, index enrichment is used.

The division of work between the two methods can be optimized in various ways. For example, only deep levels of the thesaurus can be considered for index enrichment (*matrix, equation, inequality*, ... → @*mathematics*; here we mark the introduced hypernyms with @, as mentioned in Section 0), while the upper level hierarchy, if need by the query, can be taken into account by query enrichment (*science* → @*mathematics* OR @*physics* OR @*chemistry*). What is more, the way the users most frequently modify their queries can be automatically, semi-automatically, or manually learned and implemented in the index enrichment. For instance, if the users frequently exclude the form *como* from the paradigm of the Spanish verb *comer* (see section 2.2), then it should be excluded at the stage of index enrichment. The query with the unmodified paradigm will be internally (and transparently for the user) implemented, if needed, through query enrichment as "=*comer* OR *como*" (supposing that morphologically reduced forms are marked with =).

The combined technique compensates for the query inflation problem caused by query enrichment, especially of the thesaurus-based type. It has the advantage of

higher performance due to smaller queries, without sacrificing flexibility. Obviously, it implies both advantages and disadvantages of index enrichment. Specifically, it gives the advantage of the possibility to consider the context. On the other hand, it introduces the methodological and technical disadvantages of index enrichment listed in the section 0, such as maintenance problems and undesirable coupling of the DBMS and the lingware.

9 Conclusions

We have shown that the traditional technique for non-literal string matching in information retrieval – index enrichment – has some inherent disadvantages, and have suggested a new technique – lazy query enrichment – that allows greater flexibility of queries, better overall system architecture, and easier maintenance.

Our method still has two problems: (1) the enriched queries in some cases are significantly larger and thus work slower, and (2) it is not obvious how to take the context of the keyword into account. As a solution, a combination of the query and index enrichment methods was discussed.

References

1. Aho, Alfred V. *Algorithms for finding patterns in strings*. In J. van Leeuwen (ed.), Handbook of Theoretical Computer Science, chapter 5, pp. 254-300. Elsevier Science Publishers B. V., 1990.
2. Cassidy P. *An Investigation of the Semantic Relations in the Roget's Thesaurus: Preliminary Results*. In: A. Gelbukh (ed.), Computational Linguistics and Intelligent Text Processing, IPN-UNAM, Mexico, to appear. See also Proc. of CICLing-2000, February 2000, CIC-IPN, Mexico City, ISBN 970-18-4206-5.
3. Gelbukh, A. *A data structure for prefix search under access locality requirements and its application to spelling correction*. Proc. of MICAI-2000: Mexican International Conference on Artificial Intelligence, Acapulco, Mexico, 2000.
4. Gelbukh, A., G. Sidorov, and A. Guzmán-Arenas. *Use of a Weighted Topic Hierarchy for Document Classification*, Matoušek et al., TSD-99: Text, Speech, Dialogue. Lecture Notes in Artificial Intelligence N 1692, Springer, 1999.
5. Gusfield, Dan. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997; ISBN: 0521585198.
6. Guzmán-Arenas, Adolfo. *Finding the main themes in a Spanish document*, Journal Expert Systems with Applications, Vol. 14, No. 1/2. Jan/Feb 1998, pp. 139-148.
7. Fellbaum, Ch. (ed.) *WordNet as Electronic Lexical Database*. MIT Press, 1998.
8. Frakes, W., and R. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.
9. Hausser, Ronald. *Three principled methods of automatic word form recognition*. Proc. of VEXTAL: Venecia per il Tratamento Automatico delle Lingue. Venice, Italy, Sept. 1999.
10. Koskenniemi, Kimmo. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production*. University of Helsinki Publications, N 11, 1983.
11. Kowalski, Gerald. *Information Retrieval Systems Theory and Implementation*, Kluwer Academic Publishers, 1997.
12. Lenat, D. B. and R. V. Guha. *Building Large Knowledge Based Systems*. Reading, Massachusetts: Addison Wesley, 1990. See also more recent publications on CYC project, http://www.cyc.com.

Extending the Re-use of Query Results at Remote Client Sites

Jerome Robinson Barry G. T. Lowden

Department of Computer Science, University of Essex
Colchester, Essex, CO4 3SQ, U.K.
{robij, lowdb}@essex.ac.uk

Abstract. In 'Associative Caching' each client computer keeps a copy of its query result sets in its own local database. The purpose is to reduce the size and frequency of queries to the remote data server accessed by wide-area network. For each new query the client tries to find some or all of the required data in its local collection of result sets. This is done by syntactic comparison of the new query with each previous query, to detect overlapping data sets. Attribute-Pair Range Rules are subset descriptors which the server derives from its data, for its own use in query optimisation. These subset descriptors can be further utilized to provide descriptors for each query result set. This new information enables clients to exploit their cached data for syntactically unrelated queries.

Introduction

The contribution of this paper is the extension of client cache descriptors to support syntactically unrelated queries.

In the client-server architecture for database access, one or more data servers handle requests for data retrieval and update from multiple client processes. Communication between client and server often occurs through a network, and an internet link is increasingly common. Server response time is important to system performance, and the client machine must try to hide from its users the sometimes very significant data access delays caused by network traffic congestion or overloaded servers. Client caches represent one strategy to conceal delays in distributed systems. Data retrieved from the remote data server is retained in the client's local DBMS for possible re-use in later queries.

Caching schemes *for memory hierarchies* normally use blocks of binary code as the units for caching, without knowledge of content. Semantic caching [7] or Associative Caching [11] instead regard the whole result set received from the server as the unit for caching. The client stores the result sets in a local DBMS so that *parts* of cached data sets can be selected using *local* database queries.

For example, a remote query uses $(15 \leq d \leq 45) \wedge (12 \leq h \leq 74)$ as the filter condition to select tuples, where d and h are attribute names. The result set is sent

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 536–547, 2000.

from data server to remote client. The client caches the set and labels it with the query expression $(15 \le d \le 45) \wedge (12 \le h \le 74)$. This *descriptor* label is used to identify cache containment for future queries. Traditional query containment reasoning [e.g. 1, 14] is used to detect whether some or all of the result set for each future query is contained in the cache. The syntax of a new query is compared with the expression used as the label for the cached set. Syntactic analysis is limited by the similarity of the two expressions involved in the comparison. Increasing the number of syntactic components in the cache descriptor can overcome this limitation. The server's knowledge of data subsets can be used to extend the cache descriptor, allowing containment recognition for a wider range of future queries. In other words, the server uses its knowledge of the data to provide additional information about cached sets, to make them more useful.

A new query uses the selection condition $(20 \le d \le 40) \wedge (30 \le h \le 50)$. This is obviously a subset of the cached set, and can be retrieved from that set by using the new query on the cached set in the local database. However, if another query requests *set*$(70 \le g \le 95)$ there seems to be no connection with the cached set because the new query refers to an attribute which was not mentioned in the previous query. More information is needed in order to recognize a connection. A rule of the form $A \Rightarrow B$, meaning "IF condition A is true THEN assertion B is true", would be useful, such as:

$$(70 \le g \le 95) \Rightarrow (16 \le d \le 42) \wedge (13 \le h \le 69) \qquad \ldots\ldots\ldots \textit{ rule 1}$$

where g, d and h are attribute names. The rule means "all tuples/objects with attribute g in the range (70 .. 95) will have d values in (16 .. 42) *and* h values in (13 .. 69)". This is useful information because it means the result set for the new query is in the cached set. The subset relationship is as follows: Rule $A \Rightarrow B$ asserts that all objects which satisfy condition A will also satisfy B, therefore *set*$(A) \subseteq$ *set*(B) because all A's are B's. Furthermore, the consequent in rule 1 denotes a subset of the cached set because sub-ranges select sub-sets.

Rules of the form $A \Rightarrow B$, are needed by the client cache manager, where A is the client's new query expression and B is the label on a cached set. The data server must *generate* such rules at the time it supplies the data set to be cached. It must provide the client with rules whose antecedent conditions are sufficiently general to 'match' many future queries produced by the client. To achieve this, the server can use a larger set of rules which are descriptors for subsets of *the server's* data. Descriptors for *client* cached set data are derived from these rules. The set of rules in the server must be small enough to be practically usable, but provide enough information to match a wide range of queries. The rules must be automatically derivable by the server from its data, and query-relevant rules must be rapidly discoverable from the server's set of rules, to give to the client. Histogram rule sets [17] satisfy these requirements.

The structure of the paper is as follows. After essential preliminaries in section 2, the next section explains how the server creates appropriate sets of rules describing its own data. Section 4 identifies the table-lookup process which creates descriptors for query results from this basic collection of data subset descriptors. Section 5 discusses the problem introduced by null values, and its solution.

2 Background

Rule 1 above is an example of a subset descriptor. Antecedent condition ($70 \leq g \leq 95$) *selects* a set of multi-attribute data items such as records in a database table. The consequent records the *observation* that within the selected set all elements satisfy the constraint $(16 \leq d \leq 42) \wedge (13 \leq h \leq 69)$. This means the MIN and MAX values for attribute d are observed to be 16 and 42 respectively, and for attribute h they are 13 and 69. A subset descriptor rule is therefore a ⟨selector, descriptor⟩ pair, where the antecedent condition selects a data subset and the consequent describes it. The consequent is a set of assertions about the subset.

2.1 Basic Assumptions

1. If $A \Rightarrow B$ then *set*(A) $\subseteq$ *set*(B) where *set*(X), means the set of tuples or objects with property X.

Explanation: All objects with property A also have property B, according to the rule $A \Rightarrow B$. Therefore *set*(A) must be a subset of all objects with property B.

2. If $A \Rightarrow B$ and *set*(B) $\subseteq$ *set*(C) then *set*(A) $\subseteq$ *set*(C).

This property is used when 'matching' a query condition, C, against a consequent condition, B, to infer that *set*(A) is contained in the cache.

3. If $A \Rightarrow B$ and *set*(D) $\subseteq$ *set*(A) then $D \Rightarrow B$.

The rule $A \Rightarrow B$ asserts that all objects with property A, which includes *set*(D) in this case, have property B. This transitive inference is used in antecedent lookup, to produce Outer Descriptors.

4. *set*(A $\wedge$ B) $\subseteq$ *set*(A) because conditions added to a conjunctive constraint restrict the size of the selected set of data items.

5. If $A \Rightarrow B$ then *set*(A $\wedge$ B) **=** *set*(A) because *set*(B) contains *set*(A).

This allows a cached set to be labelled with expression A instead of A $\wedge$ B used in the query to produce the set. A simpler expression can match a wider range of future queries.

6. { a(n .. m) } $\subseteq$ { a(j .. k) | $j \leq n \wedge m \leq k$ } because a sub-range selection condition identifies a data sub-set.

This property removes the need for all possible cached sets to be individually described in the server's collection of descriptors. Each cached data set inherits properties from a super-set described in the server's set descriptors, and one way to identify relevant super-sets is by the nesting of range conditions.

2.2 Two Rules of Inference

Two basic rules of inference, found in elementary logic textbooks are:

1. Union or Additive Rule

If $A \Rightarrow B$ and $A \Rightarrow C$ then $A \Rightarrow B \wedge C$.

This is the rule by which Attribute Pair rules [16] with the same antecedent condition are combined into a single multi-consequent rule.

For example,

from (a ≥ 95) ⇒ (70 ≤ g ≤ 95)
and (a ≥ 95) ⇒ (d IN { "EDBT", "MFCS"})
where a, g and d are attribute names,
produce: (a ≥ 95) ⇒ (70 ≤ g ≤ 95) ∧ (d IN { "EDBT", "MFCS"})

2. Decomposition or Projective Rule

If $A \Rightarrow B \wedge C$ then $A \Rightarrow B$ and $A \Rightarrow C$.

This rule allows a set of multi-consequent rules to be used as a lookup table for any subset of consequent conditions.

For example: from $A \Rightarrow B \wedge C \wedge D \wedge E \wedge F$ extract $A \Rightarrow C \wedge D \wedge F$.

3 Creating Subset Descriptions of Server Data

Appropriate subsets of the server's data must be chosen and described in order to provide relevant information when a new descriptor must be produced for a query result set. The antecedent condition in each selector ⇒ descriptor rule needs to match many query conditions. A single-condition antecedent is used because of its greater match potential than a two- or three-condition antecedent. (25 ≤ a ≤ 50) is a better antecedent than (25 ≤ a ≤ 50) ∧ (10 ≤ d ≤ 20) since the latter denotes only a subset of the former. This is important because antecedents become cache descriptor components whose role is to contain future queries.

The server creates rules by *partitioning a database table* (or the extent of a class in an OODB). The table may be the Join of several base relations, if that is a table from which queries select tuples. The table is partitioned using the values of a single attribute. For a numeric attribute, its *range* is divided into in N sub-ranges, where N is the number of subset descriptor rules required. Each sub-range selects a subset of tuples/objects to describe. For non-numeric attributes, individual attribute values are used as antecedents, e.g.: (h = "doctor"). If there are too many different strings then only *frequently used* values of the non-numeric attribute become rule antecedents.

For example, the server selects one column of the database table as antecedent attribute, and determines its MIN and MAX values. Partitioning this range into fifty contiguous sub-ranges will identify fifty disjoint subsets to describe. Each sub-range

selects the set of tuples whose value for the antecedent attribute is in that range. The sub-range, such as ($15 \leq a \leq 30$) for attribute 'a', could be used as selection condition in a database query, to select from the table tuples which would then be described. In practice rule derivation involves a single scan through the table data, as explained below, rather than fifty selection queries. Each subset is *described* by summarising the values it contains for non-antecedent attributes. The result of this process is a set of fifty subset descriptors, all with the same antecedent attribute. Further sets may be produced by choosing other attributes as antecedent. Choice of attributes, in consequent as well as antecedent, is based on the observed query profile.

A single scan through the table is sufficient to generate several *sets* of rules, each with a different antecedent attribute. Each tuple encountered during the scan maps to a single bucket in each rule set. There are fifty buckets for each rule set, for example, corresponding to each of the contiguous sub-ranges. A descriptor is created incrementally for each sub-set, so that at any point in the scan each of the descriptor rules describes all the tuples encountered so far which belong to its sub-set.

For example, at some point in the scan one subset descriptor has the form:

$$a(15..30) \Rightarrow c(71..94) \wedge g(101..156)$$

This indicates that all tuples encountered so far whose attribute 'a' value was in the range $15 \leq a \leq 30$ were found to have values of attribute 'c' in the range 71..94 and attribute 'g' values in 101..156. If the next tuple in the table has the following 'a', 'c' and 'g' values:

c		a		g	
96	…	29	…	135	…

then the value of the antecedent attribute 'a' maps it to the bucket with antecedent range a(15..30), the value of 'c' requires the range describing all 'c' values to increase from (71..94) to (71..96), and the value of 'g' does not change the descriptor because 135 agrees with the assertion that all 'g' values are in the range 101..156.

Each subset descriptor is an *exact* description of the data, with no exceptions, unlike rules produced by KDD [13]. They share this exact descriptor property with the subsets described in the widely used OLAP cube data structure.

The server has its own uses for the collection of rules it derives from its data. It uses them for query reformulation to provide faster query processing, as explained in [12, 15, 16, 17] for example. Deriving cache descriptors from them is therefore an extension to the value of an existing system.

4. Producing Descriptors for Cached Query Results

These are derived by the server from its collection of subset descriptors, using the consequents as a look-up table. *For example*, the following five rules describe five

disjoint subsets obtained by partitioning a table, using contiguous but disjoint sub-ranges of attribute g.

$(0 \le g < 15) \Rightarrow (31 \le b \le 64) \wedge (d \text{ IN } \{\text{“AB”, “AC”, “GH”}\}) \wedge (417 \le f \le 465)$
$(15 \le g < 30) \Rightarrow (46 \le b \le 72) \wedge (d \text{ IN } \{\text{“AC”, “AD”, “GH”}\}) \wedge (404 \le f \le 451)$
$(30 \le g < 45) \Rightarrow (59 \le b \le 93) \wedge (d \text{ IN } \{\text{“AA”}\}) \wedge (395 \le f \le 448)$
$(45 \le g < 60) \Rightarrow (75 \le b \le 107) \wedge (d \text{ IN } \{\text{“JK”, “LN”}\}) \wedge (382 \le f \le 430)$
$(60 \le g < 75) \Rightarrow (88 \le b \le 121) \wedge (d \text{ IN } \{\text{“AC”,“GH”,“JK”,“RF”}\}) \wedge (369 \le f \le 417)$

Subsets are described by specifying the range of values observed in attributes b, d and f. Each consequent is a vector of three assertions in this example. Corresponding vector elements form a column. The table composed from these columns is used as a look-up table for conjunctions of conditions which occur in new queries, since each consequent is a *conjunction* of assertions. The purpose of consequent lookup is to identify one or more antecedent conditions. Each antecedent denotes a set of tuples which is *contained* in the set which would be selected by the conjunction of conditions in the consequent. This is the property “If A $\Rightarrow$ B then *set*(A) $\subseteq$ *set*(B)”, in section 2.1 above. Antecedent sets are also contained in the cached result set of the query whose selection conditions are used for consequent lookup, because of the Decomposition rule of inference (section 2.2).

For example, the single condition $(20 \le b \le 80)$ is used in a new query whose result set will be cached at the client site. The server looks-up this condition in the *consequents* of all available rule sets which use attribute b in their consequents. A query condition ‘matches’ a consequent condition if the consequent condition denotes *a subset* of the query condition set. For *numeric* attributes this means the rule condition is a *sub-range* of the query condition range. For non-numeric attributes, such as d in the rules above, the data set selected by a condition depends on the string values specified in the condition, so *set*(A) $\subseteq$ *set*(C) if condition A contains a subset of the string values specified in condition C.

To ‘match’ the new query condition $(20 \le b \le 80)$, the b column is scanned for a sub-range of (20 .. 80). In the five rules above, the first two rules match the query condition. Since the two rules are adjacent they can be merged to describe a larger subset, *i.e.* $(0 \le g < 30) \Rightarrow (31 \le b \le 72)$. The first two rules assert that all tuples with g values in the range [0 .. 30) have b values in the range specified by the new query. Therefore those tuples will be in the cached set, so the expression $(20 \le b \le 80)$ which is used to describe the cached result set can now be supplemented with the information that *set*$(0 \le g < 30)$ is a subset.

Since the rule consequents are conjunctions of conditions they can be used to lookup the conjunctions of conditions used in new queries. *For example*, a new query whose result set will be cached is $(50 \le b \le 100) \wedge (250 \le f \le 500)$. All the five rules above match (are sub-ranges of) condition $(250 \le f \le 500)$, but only the third rule also matches $(50 \le b \le 100)$. Therefore *set*$(30 \le g < 45)$ is a subset of the cached query result set created by selection condition $(50 \le b \le 100) \wedge (250 \le f \le 500)$.

The product created by using the consequents as a look-up table is a collection of antecedent conditions, from rules whose consequents matched the new query's conditions. These antecedent expressions are all added to the descriptor for the cached set of data values produced by the new query. The original descriptor for a cached set is the selection expression Q used in the query which created the set. Q is also used for the consequent lookup process described above. Each antecedent A_i obtained from the lookup table provides a rule $A_i \Rightarrow Q$ for the client cache manager.

4.1 Inner and Outer Descriptors

Cache descriptors provide information about attributes that were not mentioned in the query which generated the cached set. Subset descriptor rules held by the data server can provide information about a new attribute in two ways, as illustrated in the following example.

A cached set was retrieved using selection condition (b = "BBC"). This condition may appear in the antecedent or the consequent of a server subset descriptor

$$(25 \le d \le 35) \Rightarrow (\ b = \text{"BBC"}\ )$$
$$(\ b = \text{"BBC"}\ ) \Rightarrow (10 \le d \le 90)$$

Two different rules here describe a connection between values in attributes d and b. The first asserts that all tuples in the universe of the database which have a value of attribute d in the range $(25 \le d \le 35)$ are in the cache because they have (b = "BBC").

There may be other d values in the cache apart from those in the subset described by the first rule. Limits for all d values that exist in the cached set are specified by the second rule. There are no values outside the range (10 .. 90) because the rule asserts that all tuples in the cache, i.e. all tuples with (b = "BBC") have $(10 \le d \le 90)$.

The first rule, which we call an *Inner descriptor*, identifies one of the subsets which are completely contained in the cached set. The second rule indicates the limits for values which can be found in the cache. This *Outer Descriptor* is useful information to support containment analysis for query expressions, since it states which values are known to be *absent* from the cache.

Antecedent lookup is used to produce Outer Descriptors for a cached set. (Consequent lookup is used, as described above, for inner descriptors). *For example*, $(20 \le g \le 50)$ is the expression used to select a new set which will be cached. The rule set with g as antecedent attribute is consulted. 'Matching' in antecedent look-up means *query condition* $\subseteq$ *antecedent condition* (in contrast to consequent look-up, where *consequent condition* $\subseteq$ *query condition*. This is because the new query must imply the antecedent in antecedent lookup, whereas the consequent must imply the new query in consequent lookup). Antecedent ranges are concatenated to enclose the query condition range. Query condition $(20 \le g \le 50)$ compared with the five antecedents in the rules set above, requires the *union* of the three rules with antecedents $(15 \le g < 30)$, $(30 \le g < 45)$ and $(45 \le g < 60)$. These subset descriptors combine to describe the larger subset required by the query condition.

Descriptors combine by unioning corresponding consequents, so:

$(15 \le g < 30) \Rightarrow (46 \le b \le 72) \wedge (d \text{ IN } \{\text{“AC”, “AD”, “GH”}\}) \wedge (404 \le f \le 451)$
$(30 \le g < 45) \Rightarrow (59 \le b \le 93) \wedge (d \text{ IN } \{\text{“AA”}\}) \wedge (395 \le f \le 448)$
$(45 \le g < 60) \Rightarrow (75 \le b \le 107) \wedge (d \text{ IN } \{\text{“JK”, “LN”}\}) \wedge (382 \le f \le 430)$

combine to produce:

$(15 \le g < 60) \Rightarrow$
$(46 \le b \le 107) \wedge (d \text{ IN } \{AC, AD, GH, AA, JK, LN\}) \wedge (382 \le f \le 451).$

Set$(15 \le g < 60)$ contains *set*$(20 \le g \le 50)$, so all consequent assertions (limits to possible values) apply to that cached set.

If the cached query is a *conjunction* of conditions such as $(20 \le g \le 50) \wedge (250 \le h \le 500)$ then the rule sets for antecedents g and h are separately consulted, to produce a rule from each, such as:

$(15 \le g < 60) \Rightarrow$
$(46 \le b \le 107) \wedge (d \text{ IN } \{AC, AD, GH, AA, JK, LN\}) \wedge (382 \le f \le 451)$
derived above, and

$(250 \le h < 550) \Rightarrow (59 \le b \le 141) \wedge (d \text{ IN } \{AA, GH, KK\}) \wedge (214 \le f \le 403)$

produced in a similar way from the rule set with h as antecedent attribute. The ranges and sets for corresponding attributes in the *consequents* of the two rules *necessarily intersect* because overlapping sets must have something in common. Conversely, if the descriptors for any single attribute fail to intersect it means that the query conditions, $(20 \le g \le 50)$ and $(250 \le h \le 500)$ in this case, select *disjoint subsets*. So their conjunction in a query selects the empty set; there is nothing to cache and no need merge rule consequents in this way to create a cache descriptor.

Corresponding attribute descriptors for the two rules are unioned to specify extreme value limits for that attribute in the cached set:

$(20 \le g \le 50) \wedge (250 \le h \le 500) \Rightarrow$
$(46 \le b \le 141) \wedge (d \text{ IN } \{AC, AD, GH, AA, JK, LN, KK\}) \wedge (214 \le f \le 451).$

This is the Outer Descriptor for the cached set, selected by the antecedent condition.

4.2 Partial Containment

A client is interested to know whether *part* of the data which will result from its next query can be found in its local cache. It will use that part of the result set immediately, and create a *remainder query* to get the rest of the data from the server.

For example, if a new query wants data selected by condition $(27 \le d \le 114)$, and one of the subsets known to be present in a cached set (because it is one of the expressions in the extended descriptor of a cached set) is *set*$(20 \le d \le 40)$, then the *set*$(27 \le d \le 40)$, can be selected from the cache, and the remainder query to the server will use selection condition $(40 < d \le 114)$ instead of $(27 \le d \le 114)$.

For a cached set labelled $(15 \le d \le 45) \wedge (12 \le h \le 74)$, if a new query requests data matching selection condition $(20 \le d \le 40)$, then *some* of the required data can be found in the cache database because it contains $set((15 \le d \le 45) \wedge (12 \le h \le 74))$, which in turn contains $set((20 \le d \le 40) \wedge (12 \le h \le 74))$. This latter set, is a subset of the required data, because $set((20 \le d \le 40) \wedge (12 \le h \le 74)) \subseteq set(20 \le d \le 40)$ according to basic assumption 4 in section 2.1 above. The remainder of the data for the new query is fetched from the remote server. The remainder query will request tuples WHERE $((20 \le d \le 40)$ AND NOT $(12 \le h \le 74))$. The benefit of partial containment by the cache is *immediate access* to some of the data, hiding the internet latency delay before the rest of the result set arrives.

5. Null Values

The discussion above must now be extended to handle null values. A *null value* is an empty field in a data object such as a tuple. This affects the meaning of a rule such as $(114 \le c \le 129) \Rightarrow (20 \le a \le 40)$, whose consequent describes the set selected by the antecedent condition. All tuples in that set *which have a value for attribute 'a'* will have $(20 \le a \le 40)$. But if null values exist in attributes used in rules, additional information about those nulls is required to prevent incomplete query result sets from the cache. The problem introduced by nulls in the present context is illustrated by the following example.

A query selects tuples using the condition $(20 \le a \le 40)$, and the result set is cached. Any tuples with an empty field for attribute 'a' will not be selected since they do not satisfy the condition. So a tuple with three fields for attributes a, b and c: ⟨ NULL, 61, 120 ⟩ respectively, will not be selected, for example. But a cache descriptor rule: $(114 \le c \le 129) \Rightarrow (20 \le a \le 40)$ *which ignored null values* would assert that all database tuples with $(114 \le c \le 129)$ are in the cache. A new query requesting tuples with (c = 120) would therefore only look in the cache, so it would not get tuples such as ⟨ NULL, 61, 120 ⟩ where c = 120 and a = NULL, which would be obtained if the new query had gone directly to the data server instead of the cache.

5.1 Solving the Problem

If an attribute has null values, there is a set of tuples with a null value in that field. That set can be described, using a subset descriptor:

$$(a = \text{NULL}) \Rightarrow (83 \le c \le 147) \wedge (f \ge 147) \wedge (k \text{ IN } \{ \text{“VLDB”, “KRDB”}\} \wedge \text{etc.}$$

A single subset descriptor of this kind is produced for each antecedent *attribute*, and used as the following examples indicate:

The server receives a query requesting all tuples which satisfy the selection condition $(12.50 \le e \le 15.00)$. To derive relevant cache descriptors from its set of subset descriptors, the server does a lookup operation to match this condition in the

consequents of all available multi-consequent rule sets (for all antecedent attributes other than attribute e used in the query). There is one *rule set* per attribute, because a different attribute is used in the antecedent of each set. Suppose the rule set with attribute 'a' as antecedent provides a rule:

$$(75 \le a \le 100) \Rightarrow (12.60 \le e \le 14.05).$$

The consequent "matches" the query condition because $(12.60 \le e \le 14.05)$ is a sub-range of query condition $(12.50 \le e \le 15.00)$, and therefore

$$set(75 \le a \le 100) \subseteq set(12.60 \le e \le 14.05) \subseteq set(12.50 \le e \le 15.00).$$

If attribute e contains null values, the server consults the (e = NULL) rule. If it finds $(e = NULL) \Rightarrow (15 \le a \le 71)$, for example, then nulls do not affect the present subset. All nulls are in tuples with $(15 \le a \le 71)$ so $set(75 \le a \le 100)$ is not affected.

But if $(e = NULL) \Rightarrow (15 \le a \le 81)$ there is range overlap with $(75 \le a \le 100)$, so $set(75 \le a \le 100)$ retrieved from cached $set(12.50 \le e \le 15.00)$ would be incomplete. The server must now provide information to the client which will allow it to construct a *remainder query* for those members of $set(75 \le a \le 100)$ which have been omitted from the cache because of null values in attribute e.

The remainder query is obviously $(75 \le a \le 100) \wedge (e = NULL)$ so the client only needs to know that nulls in attribute e affect $set(75 \le a \le 100)$ within cached set $set(12.50 \le e \le 15.00)$. Therefore the cache descriptor information sent to the client, saying that $set(75 \le a \le 100)$ is a subset of the cached query set, is simply tagged with the comment, "nulls in e".

This method of managing null values applies equally to conjunctive queries. *For example*, a new query to the server requests the tuples which satisfy condition

$$(12.50 \le e \le 15.00) \wedge (b = \text{"KJ\_4/OCX"}) \wedge (d \ge 1/7/97).$$

Table look-up in the multi-consequent rule set for antecedent attribute 'a', using the Projective Rule from Section 2.2, produces the descriptor:

$$(75 \le a \le 100) \Rightarrow (12.60 \le e \le 14.05) \wedge (b = \text{"KJ\_4/OCX"}) \wedge (d > 1/7/98).$$

The null rule for each consequent attribute is consulted:

$$(e = NULL) \Rightarrow (15 \le a \le 81)$$

no nulls for attribute b

$$(d = NULL) \Rightarrow (50 \le a \le 150)$$

The information to the client for the Inner Descriptor $(75 \le a \le 100)$ associated with cached set, $set((12.50 \le e \le 15.00) \wedge (b = \text{"KJ\_4/OCX"}) \wedge (d \ge 1/7/97))$ therefore states "nulls in d and e", so that the client will generate remainder query

$$(75 \le a \le 100) \wedge ((d = NULL) \vee (e = NULL))$$

to retrieve the remainder of $set(75 \le a \le 100)$ which is not in the cache.

6 Conclusions

The benefits of caching query result sets, rather than pages, at client sites were demonstrated in [4, 5]. These benefits are extended if the client can also use its cache to answer syntactically unrelated queries. The contribution of this paper is a system which extends cache descriptors to support syntactically unrelated queries.

In previous work [12, 15, 16, 17] on 'semantic query optimisation' [10, 18, 19, 20] the authors developed a particular type of subset descriptor, which the data *server* used for *query optimisation* (rewriting the query in a form that is answerable more quickly). The current paper extends the use of rules for query optimisation, by utilizing descriptors in the Server to provide *new* subset descriptors for remote client caches, and so extends current research on 'associative caching' [e.g. 4, 5, 11] and other applications of Materialized Views.

Caches at remote client sites in a wide-area network convert a centralized data server into a distributed system with data sets nearer the place they are used. This introduces benefits and disadvantages associated with distributed databases. Many researchers are working on the difficulties and problems introduced by associative client caching, such as cache consistency maintenance and reduced data availability through aggregation and projection [e.g. 4, 9, 11, 21]. Solutions are being reported for the various problems. But all systems so far considered have the limitation that new queries must possess syntactic similarity with cached queries. Useful data may be available locally for a new query but its existence is not recognized because the new query specifies that data in a different way. Extending the descriptor for a cached set will improve its capture potential for future queries and therefore extend the ability to re-use cached data for later queries.

This paper concentrated on queries that *select* data subsets. Other queries go beyond this in *describing* the selected data. The result set of a query that uses aggregate functions is a set of subset descriptors. Each result tuple from a GROUP BY query describes a subset, for example. Users of aggregate queries tend to frequently use similar aggregate queries. This explains the effectiveness of the pre-computed 'cube' data structure (collection of subset descriptors) used in multi-dimensional databases. The server's pre-computed collection of subset descriptors discussed in section 3 is analogous to a cube. It seems likely, therefore, that these can be shared with clients to accelerate aggregate query answering, and this is the subject of our current research.

References

1. Adali, S., Candan, K. S., Papakonstantinou, Y., Subrahmanian, V. S.: Query Caching and Optimization in Distributed Mediator Systems. Proc ACM SIGMOD Conf. (1996) 137-148.
2. Amsaleg, L., Bonnet, P., Franklin, M. J., Tomasic, A. and Urhan, T.: Improving Responsiveness for Wide-Area Data Access. Data Engineering Bulletin 20(3): 3-11 (1997).

3. Arens, Y., Knoblock, C. A.: Intelligent Caching: Selecting, Representing, and Reusing Data in an Information Server. Proc. CIKM'94, Third International Conference on Information and Knowledge Management (1994) 433-438.
4. Basu, J., Poess, M. and Keller, A. M.: High Performance and Scalability Through Associative Client-Side Caching, Seventh International Workshop on High Performance Transaction Systems, Pacific Grove, CA, September 1997.
5. Basu, J., Poess, M. and Keller, A. M.: Performance Analysis of an Associative Caching Scheme for Client-Server Databases, Technical Note STAN-CS-TN-97-61, Stanford University, Computer Science Dept., September 1997.
6. Bonnet, P., Tomasic, A.: Partial Answers for Unavailable Data Sources. Proc. FQAS'98, Third International Conference on Flexible Query Answering Systems (1998) 43-54. (LNCS 1495).
7. Dar, S., Franklin, M. J., Jonsson, B. T., Srivastava, D., Tan, M.: Semantic Data Caching and Replacement, Proc. 22nd VLDB Conference (1996) 330-341.
8. Franklin, M., Kossmann, D.: Cache Investment Strategies. Univ. of Maryland Technical Report CS-TR-3803/UMIACS-TR-97-50, May, 1997.
9. Hsu, C-N., Knoblock, C. A.: Discovering Robust Knowledge from Databases that Change. Journal of Data Mining and Knowledge Discovery, 2, 1-28 (1998).
10. Hsu, C-N., Knoblock, C. A.: Semantic Query Optimization for Query Plans of Heterogeneous Multi-Database Systems. IEEE Transactions on Knowledge and Data Engineering, accepted for publication, 1999.
11. Keller, A. M., Basu, J.: A Predicate-based Caching Scheme for Client-Server Database Architectures. VLDB Journal 5(1) 1996, 35-47.
12. Lowden, B.G.T., Robinson, J., Lim, K.Y.: A Semantic Query Optimiser using Automatic Rule Derivation. WITS'95, 5th Intl. Workshop on Information Technologies and Systems (1995) 68-76.
13. Piatetsky-Shapiro, G.: Discovery, Analysis and Presentation of Strong Rules, Knowledge Discovery in Databases, Eds. G. Piatetsky-Shapiro and W. J. Frawley, MIT Press (1991) 229-248.
14. Qian, X.: Query Folding. 12th IEEE Intl. Conference on Data Engineering (1996) 48-55.
15. Robinson, J., Lowden, B. G. T.: Data Analysis for Query Processing. 2nd Intl. Symposium on Intelligent Data Analysis (1997) 447-458. (LNCS 1280)
16. Robinson, J., Lowden, B. G. T.: Attribute-Pair Range Rules. Proc. DEXA'98, 9th Intl. Conference on Database and Expert Systems Applications (1998) 680-691. (LNCS 1460)
17. Robinson, J., Lowden, B. G. T.: Semantic Query Optimisation and Rule Graphs. KRDB'98, 5th International Workshop on Knowledge Representation meets Data Bases (1998).
18. Shekhar, S., Hamidzadeh, B., Kohli, A. and Coyle, M.: Learning transformation rules for semantic query optimization: A data-driven approach, IEEE Transactions on Knowledge and Data Engineering, 5(6), 950-964, 1993.
19. Shenoy, S.T., Ozsoyoglu, Z.M.: A System for Semantic Query Optimization, Proc ACM SIGMOD Conference, 1987, pp 181-195
20. Siegel, M., Sciore, E. and Salveter, S.: A Method for Automatic Rule Derivation to Support Semantic Query Optimization, ACM TODS 17(4) 563-600, 1992.
21. Srivastava, D., Dar, S., Jagadish, H. V. and Levy, A. Y.: Answering Queries with Aggregation Using Views, Proc. 22nd VLDB Conference (1996) 318-329.

Discovering and Representing InterSchema Semantic Knowledge in a Cooperative Multi-Information Server Environment

A-R. H. Tawil, W. A. Gray, and N. J. Fiddian

Department of Computer Science, Cardiff University, U. K.
{Abdel-Rahman.Tawil, N.J.Fiddian, W.A.Gray}@cs.cf.ac.uk

Abstract

Discovering interschema semantic knowledge between corresponding elements in a cooperating Multi-Information Server (MIS) environment requires deep knowledge, not only about the *structure* of the data represented in each server, but also about the commonly occurring differences in the intended *semantics* of this data. The same information could be represented in various incompatible structures, and more importantly the same structure could be used to represent data with many diverse and incompatible semantics. Interschema semantic knowledge can only be detected if both the structural and semantic properties of the schemas of these servers are made explicit and *formally represented* in a way that a computer system can process. Unfortunately, very often there is lack of such knowledge and the local schemas, being semantically weak as a consequence of the limited expressiveness of traditional data models, do not help the acquisition of this knowledge. The solution to overcome this limitation is primarily to upgrade the semantic level of the IS local schemas through a semantic enrichment process by augmenting these local schemas to semantically enriched schema models, then to use the enriched schema models in detecting and representing correspondences between classes belonging to different schemas. In this paper we investigate the possibility of using domain ontologies both for building semantically rich schema models, and for expressing interschema knowledge and reasoning about it. We believe that the use of domain ontologies in this setting has two important advantages. On the one hand, it enables a semantic approach for interschema knowledge specification, by concentrating on expressing conceptual and semantic correspondences between both the conceptual (intensional) definition and the set of instances (extension) of classes represented in different schemas. On the other hand, it is exactly this semantic nature of our approach that allows us to devise reasoning mechanisms for discovering and reusing interschema knowledge when the need arises to compare and combine it.

Keywords: Semantic interoperability, mediators, domain ontologies, schema enrichment, compositional modelling.

1 Introduction

Virtually all of the cooperative MIS approaches proposed in the literature fall into one of two categories: the *tightly-coupled* approach [3, 1], where the database

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 548–562, 2000.

administrator is responsible for the creation and maintenance of the integrated schema, or the *loosely-coupled* approach [12, 13], where it is the user's or system's responsibility to create the integrated schema view(s). Recently, a number of research efforts have focused on the various aspects of *mediating knowledge*, which holds more extensive semantic information than other integration approaches. The *mediator* approach [2, 16, 15] provides seamless access to a collection of related, but possibly heterogeneous and distributed ISs by constructing semantically rich integrated views of the underlying information sources to which access is required. The notion of interschema knowledge is critical for the development of any of these approaches. In the case of a tightly-coupled approach [4, 17], where one or more shared schemas are used to encapsulate the underlying conflicts, interschema semantic knowledge provides the necessary information for building the shared (global) schema. With the loosely-coupled and the information mediator approaches, where it is the *user's* or *intelligent system's* responsibility to detect and reconcile semantic conflicts, interschema semantic knowledge is used for understanding the contents of different databases so as to share relevant information.

1.1 Related Work and Paper Contribution

As the process of identifying semantically similar schema elements requires knowledge of the semantics of the data, it is no surprise that most of the proposed cooperative IS approaches advocate the use of semantic data models (e.g. *functional*, *object-oriented*, *frame-based*). In particular the mediator approach advocates the use of Knowledge Representation Systems (KRSs), particularly those descendent of the KL-One [24] family of Knowledge Representation Languages (KRLs), also known as concept languages or Description Logic (DL) languages, and uses these KRLs to build rich semantic models. For example, the LOOM knowledge representation system was used in the SIMS project [2] to describe an ontology [1] of the transport domain. Classic description logic was used in the Observer project [16] to describe ontologies and terminological relationships between concepts. CARIN, a dialect of description logic was used in the Information Manifold (IM) project [15] for describing IM *world-view* concepts. However, description logic and KL-One style languages are not the only languages used for building semantic data models. In the *context interchange project* [9], F-Logic [11] is used to describe ontologies, and in [10] Prolog is used in a bottom-up approach to ontology construction. It is not clear to date which requirements a language for semantic modelling should satisfy.

In this paper we advocate the use of an intermediate Generic Knowledge Model (GKM) that is DBMS Data Definition Language (DDL) and application-specific semantics *independent*. The goal of the intermediate model is to provide a logical definition of the constructs used in the representation of *structural* and

[1] A formal specification of the concepts and relationships that hold among these concepts.

semantic knowledge of the IS schemas which formally and consistently defines their meaning within the representation. The GKM establishes a *base vocabulary* which serves as an implicit *interlingua* between the heterogeneous ISs and is implemented as part of a rich semantic based integration system for a medical application, called *MetaMed*. We provide a framework for enriching the individual IS schemas with semantic domain knowledge (formulated in terms of the GKM vocabulary) to make explicit the assumptions which may have been made by the designer, are of interest to the integrator (interpreter or user), and which may not be captured using the DDL language of that server. The enriched semantic knowledge of the individual ISs is organised by levels of schematic granularity: *database*, *schema*, *attribute* and *instance* levels, giving rise to semantically rich schema models. This provides the basis for discovering and formally representing interschema semantic knowledge in an enriched representation of the databases, and allows for accessing and integrating data respective to each IS.

The main contribution of our work is therefore focused on providing the basis for a semantic and logical approach to the problem of discovering and representing interchema knowledge. We conjecture that interoperability between sets of heterogeneous ISs is best achieved by concentrating on expressing conceptual and semantic correspondences between both the conceptual (intensional) definition and the set of instances (extension) of classes represented in different schemas. Moreover, our approach of organising semantic knowledge into *database*, *schema*, *attribute* and *instance* levels turns the focus onto the discovery of interschema semantic knowledge that can be useful in a cooperative environment. This is in contrast with many techniques, which focus on accessing (integrating) the data, with the discovery phase acknowledged as having been completed prior to the application of the technique [3, 12]. On the other hand, several recent papers in the literature share our general goal of representing and expressing interschema knowledge. For example, Navathe and Elmasri [14] address the problem of attribute equivalence; Siegle and Madnick [20] enrich the attribute domain semantics with metadata, thereby providing a better semantic mapping of corresponding attributes; Collet and Huhns [8] use the Cyc knowledge representation system for specifying and reasoning about the interrelationships between classes of objects in different sources. These approaches differ from ours in that only extensional interdependencies between classes are considered, and interschema knowledge is not explicitly represented. In [6, 18] a logical approach similar to ours is used for both expressing interschema knowledge and reasoning about it.

The main difference with our work is that while they express semantic interdependencies between classes belonging to different information systems based on a simple mapping relationship between these classes and a shared area, we are more concerned with expressing semantic interdependencies between the descriptions of the classes belonging to different information systems. We model the content of the information sources as *composite concepts* (descriptions) formulated in terms of the predefined semantics of the shared area, then use this

knowledge to express semantic interdependencies between the *description* of the classes belonging to different information systems. Hence, our approach provides more extensive semantic knowledge about these classes, which proved to be necessary for discovering and reusing interschema knowledge when the need arises to compare and combine it.

The remainder of this paper is organised in the following way. In the next section we discuss how the MGKM is built. In section 3, we examine the schema enrichment process and give an example of a semantically enriched schema model. In section 4, we describe a model for representing interschema knowledge between classes belonging to different databases, and discuss the associated reasoning mechanisms required for discovering and representing such knowledge.

2 The MetaMed Generic Knowledge Model (*MGKM*)

The retrieval of desired information dispersed among MISs requires general familiarity with their contents and structure, with their query languages, with their location on existing networks, and more. This, in turn, requires that the semantic model provide the necessary "*vocabulary*" (*terms* or *concepts*) and language flexibility for describing such knowledge. Consider that we are particularly interested in capturing the *structural* and the *semantic* interdependencies between classes of the individual ISs. The GKM should provide the necessary body of knowledge for capturing and representing the semantic and structural details of these servers. In the MetaMed GKM (MGKM), this is achieved by means of an ODMG-93 based knowledge model accompanying the semantic knowledge model of the problem domain. We adopt *a subset* of the Object Definition Language (ODL) specification based on the ODMG standards [7], called ODL_{I^3} [2], as our GKM for addressing representational heterogeneity in the metadata of schema structures. We also advocate the use of a semantic model of the federation or application domain as our GKM for addressing the heterogeneity of metadata Real World Semantics (RWSs). Currently we are using a master notation of medical terminology and COding REference (CORE) model, called GALEN [3] (Generalised Architecture for Languages, Encyclopaedias and Nomenclatures), as the *semantic model* used to support the integration of medical information servers. The Ontolingua OKBC Knowledge Representation Language (KRL) is the modelling formalism with which both the MetaMed ODL ($MODL_{I^3}$) and MetaMed GALEN (MGALEN) knowledge models are built.

Both the MODL and the MGALEN knowledge models of the MGKM express conceptualisations which are specific for particular domains (the *ODL* and the *medical* domains) and hence are considered as domain ontologies.

[2] Derived from the I^3 mediator language proposal, as described in [5].

[3] GALEN is a EU-funded project as part of the Advanced Informatics in Medicine programme of the Commission of the European Union, AIM 2012 [19].

2.1 A Shared Medical Model of the Application Domain

The MGALEN domain model includes a hierarchical terminological knowledge base with nodes representing all objects, actions and states possible in the domain. In addition, it includes indications of all relationships possible between classes in the model. In MGALEN the primary breakdown is into: *Generalised-Structures* – abstract or physical things with parts independent of time (e.g. cell, leg, virus); *Generalised-Substances* – continuous abstract or physical things independent of time (e.g. radiation, urine, drug); *Generalised-Processes* – changes which occur over time (e.g. breathing, clotting, irradiation) ; and *Modifiers* – a heterogeneous grouping of adjectives and qualifiers (feature, state, selector, or role).

A taxonomy of *domain relations* is influenced by, and supports, the concept hierarchy (concept taxonomy) outlined above. It includes indications of all relationships possible between concepts in the model. Here, the primary distinction is between:

- *Constructive-Relations*, linking processes, structures, and substances together (e.g. Is-Suffering-From, Is-Suspected-From, Holds-Information-About, Has-Location, etc), and
- *Modifier-Relations*, linking processes, structures, and substances to modifiers (e.g. Has-Absolute-State, Has-Trend-In-State, Has-Pathological-Status, etc).

Domain relations (also referred to as *slots* or *roles*) are used to describe the *composition* (relationship) between two concepts or two concept values. A domain relation can be seen as a predicate with two arguments: the *domain* (the concept with which it is associated) and the *range* (the type of values that it can have). This compositional approach allows for detailed description of complex concepts while preserving the structure provided by the individual basic concepts in the description. In addition, it provides a mechanism for representing descriptive knowledge using the defined vocabulary. For example, the Right-Lung-Pneumococal-Infection concept is a composite concept composed from both the Right-Lung and Pneumonia concepts. Moreover, Right-Lung is a composite concept composed from both the Lung concept and the Right selector concept, and Pneumonia is the name of an Infection located in the Lung. Hence, the following KIF axioms[4] hold:

```
(=> (Patient ?X)
   (Subclass-Of ?X Person)).

(=> (and (Infection ?X)
         (Has-Location ?X Lung))
    (Pneumonia ?X)).
```

```
(=> (and (Lung ?X)
         (Has-Laterality Lung Right))
    (Right-Lung ?X)).

(=> (and (Pneumonia ?X)
         (Has-Location ?X Right-Lung))
    (Right-Lung-Pneumococal-Infection ?X)).
```

[4] The internal representation of every Ontolingua OKBC ontology is always expressed as a set of KIF axioms.

2.2 A Shared MODL$_{I^3}$ Knowledge Model of Schema Structures

In order for any collaboration to take place among semantically similar schema elements of disparate and structurally heterogeneous information servers, a *common data model* and a *common data definition language* for describing the structure of the sharable data must be adopted. The MODL$_{I^3}$ knowledge model allows for the declarative specification of ODMG schema structures based on the OKBC representational language and is capable of expressing the representational semantics of the schema structures of a variety of data models and Data Definition Languages (DDLs) (e.g. characteristics of relations including their attributes, primary keys and foreign key relationships). It is however unable and is not intended for capturing the actual semantics of the schema contents. The MODL$_{I^3}$ knowledge model is a much more restricted and confined model compared to the MGALEN semantic model. In MODL$_{I^3}$ the primary breakdown is into: Logical-Structure (e.g. odl-schema); *Primitive-Value-Type* (e.g. boolean, character, string); and *Odl-Definition* (e.g. interface, attribute, extent).

An MODL$_{I^3}$ relationship can be seen as a predicate with two arguments: the *domain* (the ODL$_{I^3}$ concept with which it is associated) and the *range* (the type of ODL$_{I^3}$ values that it can have). For example, the Has-Schema-Class relationship has `ODL-Schema` as its domain and `Interface` as its range, the Has-Class-Foreign-Key relationship has `Interface` as its domain and `Attribute` as its range.

3 The Schema Enrichment Process

One of the key difficulties during the analysis phase of schema integration is to identify semantically similar schema elements. This is due to the lack of knowledge about integrated schema semantic contents. Also, the semantics in the local schema metadata are insufficient to direct this identification process. Our schema enrichment process relates the local schema elements to the MGKM concepts that they denote. Thus an interpretation is provided, *in the form of descriptive knowledge*, for each local schema element, describing it to our system.

In the first step of our schema enrichment process, the MetaMed *schema extraction tool* extracts the metadata of the schema to be enriched and automatically creates an MODL$_{I^3}$ description of that schema. Part of the MODL$_{I^3}$ logical model of the Lung-Infection-Database schema shown in Figure 1 is the following:

```
(ODL-Schema  Lung-Infection-Database).
(Interface  Pnem-Patient).
(Extent  Pnem-Patients).
(and (Attribute  Id) (Attribute  Name) (Attribute  Consultant) ..).
(Has-Schema-DataModel  Lung-Infection-Database  Relational).
(Has-Schema-Location   Lung-Infection-Database
                       http://www.cs.ac.uk/Lung-Infection-DB.html).
(and (Has-DBMS  Lung-Infection-Database  Oracle)
```

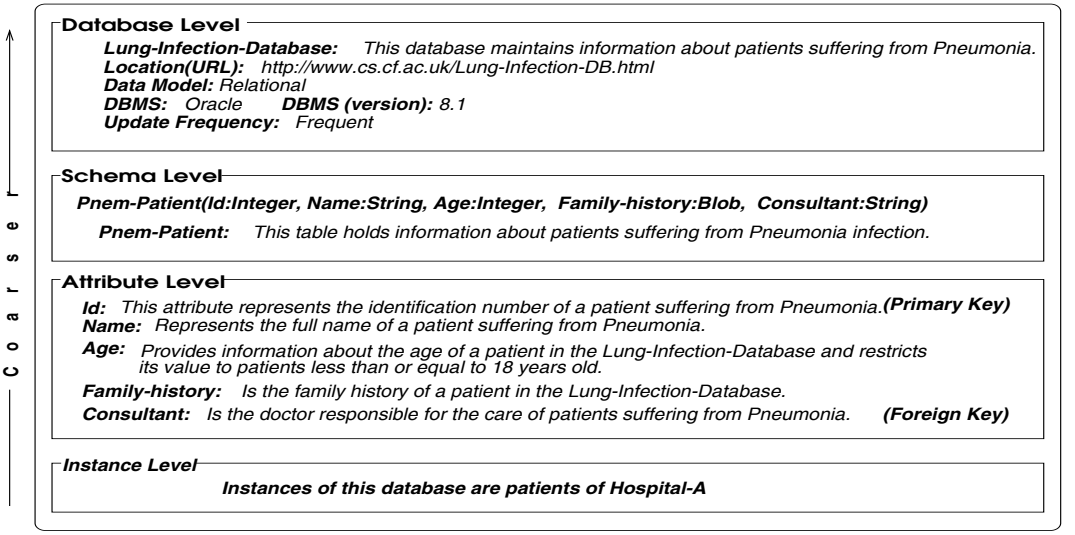


Fig. 1. A Semantic View of the Lung-Infection-Database

```
     (Has-DBMS-Version  Lung-Infection-Database  8.1)).
...
(Has-Schema-Class   Lung-Infection-Database   Pnem-Patient).
(Has-Class-Extent  Pnem-Patient  Pnem-Patients).
(Has-Primary-Key  Pnem-Patient  Id).
(Has-Foreign-Key  Pnem-Patient Consultant).
...
(and (Has-Class-Attribute   Pnem-Patient  Id)
     (Has-Attribute-Type  Id  Integer)).
...
```

In the next step of the enrichment process, the MGALEN semantic model is used to provide a semantic description of the information stored in the IS, and the semantic knowledge acquired is organised by levels of schematic granularity. This second step can not be fully automated by the system and requires the intervention of the user. However, the accuracy of the system in discovering semantic relationships between the schema terms and the MGALEN semantic model increases as the level of schematic granularity becomes finer. The system uses the knowledge acquired at the previous level to discover knowledge at the next level. MetaMed provides a graphical user environment to assist users during this phase of the enrichment process.

□ The *database level* is general – it is knowledge that pertains to all levels of schematic granularity. It includes *general domain knowledge* about the nature of the information stored in the database, as well as additional *general associated knowledge* about the domain of the types represented in the database. In

our Lung-Infection-Database example Figure 1, the general domain knowledge could simply be the knowledge that Lung-Infection-Database stores information about patients suffering from lung diseases. The associated knowledge may be something like 'lung infection is an infection located in the lung'. Hence, the Odl-Schema concept Lung-Infection-Database may be enriched by adding to it the following axioms (rules), described in the KIF formal specification language: *the Lung-Infection-Database maintains information about patients suffering from lung infection.*

```
(and (Holds-Information-About Lung-Infection-Database Patient)
     (Is-Suffering-From Patient Lung-Infection)).
```

User confirmation is required to assert this knowledge. However, once the following *general domain knowledge* is asserted, the following *general associated knowledge* is automatically added to the knowledge base of the schema model: *Lung infection is an infection which is located in the lung, and is suspected from patients having a history of smoking.*

```
(=> (and (Infection ?Inf)
         (Is-Located-In ?Inf Lung))
    (Lung-Infection ?Inf)).
```

□ The *schema level* includes an abstract (general) view of the cross-product of the domains of the attributes in a particular database schema. Accordingly the Pnem-Patient schema concept may be enriched by adding the general domain knowledge that Pnem-Patient holds information about patients suffering from pneumonia. Of course, the general associated knowledge that a *patient* is a *person* or that *pneumonia* is an *infection located* in the *lung* is automatically added to the schema model.

```
(and (Instance-Of Pnem-Patient Patient)
     (Is-Suffering-From Pnem-Patient Pneumonia)).
```

□ The *attribute level* holds the semantic domain knowledge specific to each attribute in a particular schema, as well as additional constraint knowledge about the nature of the information corresponding to each attribute as represented in its respective schema. For example, the Age attribute of the Pnem-Patient schema represents the ages of patients suffering from pneumonia, and is constrained to patients who are less than or equal to 18 years old. To enrich the Age attribute, we first define it as a *subrelation* of the Has-Person-Age relation defined with the Person concept in the MGALEN knowledge model and then restrict its value to patients less than or equal to 18 years old.

```
(Subrelation-Of  Pnem-Patient Age Has-Person-Age).
(=> (and (Pnem-Patient ?Pat)
         (Age ?Pat ?Age-Value))
    (Less-Than-Or-Equal (?Age-Value 18))).
```

□ Integratable schema concepts which are intensionally similar at the *database*, *schema* or *attribute* levels, may represent different real-world objects at the *instance (extension)* level. The asserted patient related instances of different

databases may not represent the same *real-world* objects. For example, if the set of objects that are instances of the Pnem-Patient table of the Lung-Infection-Database are patients of hospital (Hospital-A), another database may record the same information about a different set of patients in another hospital (Hospital-B). This *instance* level semantic knowledge can be asserted by adding it to the schema concept, as it affects all instances of that schema:

```
(=> (and (Instance-Of Hospital-A Hospital)
         (Pnem-Patient ?Pat)
         (Extent ?Pat ?Pat-Extent))
    (Patient-Of ?Pat-Extent Hospital-A)).
```

This asserted fact states that the set of instances (extension) of the Pnem-Patient table of the Lung-Infection-Database database refers to patients of Hospital-A.

4 Representation of InterSchema Knowledge

Irrespective of the cooperative IS approach taken, designers are faced with the problem of comparing the structure and the semantic content of the various ISs concerned to determine their interschema semantic relationships. It is important to know to what extent the participating ISs share related semantics and it is equally important to instruct the integration system about such commonalities to effectively exploit, manipulate and reason about these semantics. The role of the MGKM in this setting is to provide a unified view of the data stored in the databases of different ISs, and to organise semantic knowledge about these servers into concepts and relationships among concepts, based on the four levels of schematic granularity. The MGKM constitutes a reference knowledge model to understand the information spread over different schema models, which can be exploited in identifying interschema semantic relationships between the semantically enriched schema elements.

4.1 Discovering and Representing Interschema Assertions

Discovering interschema semantic knowledge among corresponding elements in a co-operating MIS environment requires deep knowledge about the *intensional* and *extensional* definitions associated with different database schemas. The MetaMed Identification System (MId-Sys) identifies types of semantic similarity by analysing the enriched schema knowledge, *acquired during the enrichment process*, at progressively more refined levels of schematic granularity. The rich semantic knowledge associated with the IS schema at each level guides the MId-Sys during the identification process by corroborating or contradicting its hypotheses about the identified types of semantic similarity. Finally, an identified semantic relationship between two elements of different databases is represented as a corresponding interschema assertion.

To give a feeling for the identification process we will focus on the analysis of the database schemas Lung-Infection-Database and the Cancer-Database. Both databases are partially defined in Figures 1 and 2, respectively. We assume that both schemas are enriched with semantic knowledge extracted from the MGKM and formulated in

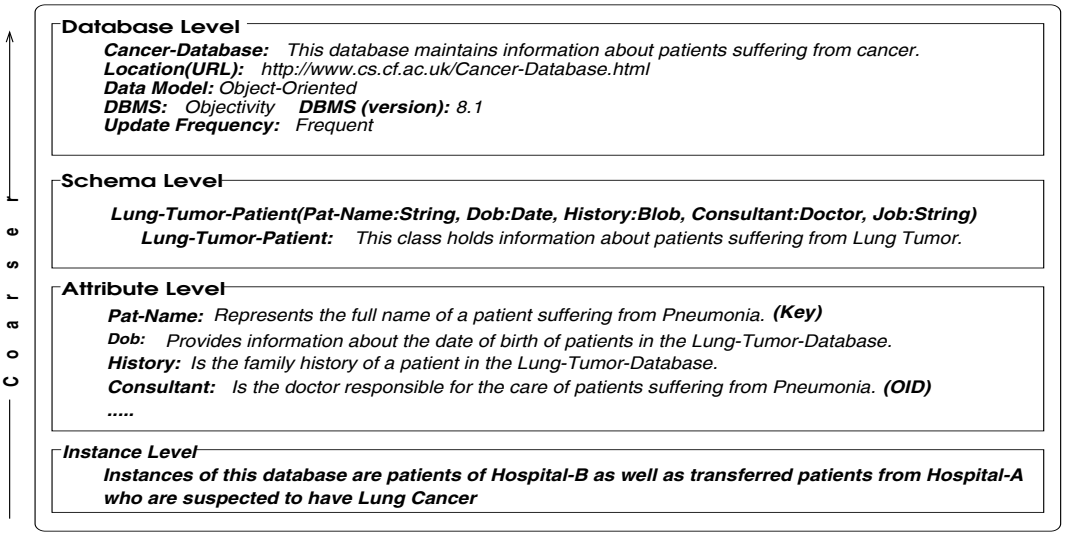


Fig. 2. A Semantic View of the Lung-Tumor-Database

terms of composite concepts as described in Section 3 of this paper.

The database schemas are first examined to see if a common or shared conceptualisation can be identified at the database level, based on the enriched knowledge acquired at that level. Using the MGKM and the descriptive schema knowledge of the individual IS schemas, the MId-Sys detects that the description of both database concepts Lung-Infection-Database and Cancer-Database holds information about patients, see Figure 3 (B). The Patient concept is the value associated with the Holds-Information-About relation of both database concepts. The integration system also detects that the concept Pneumonia[5] of the Lung-Infection-Database is actually a composite concept incorporating both the Lung and Infection concepts, and that the Cancer and Infection concept values of the Is-Suffering-From relationship are both kinds of pathologies, see Figure 3 (A). Accordingly, an *abstraction level incompatibility conflict* is identified at this level[6]. Both database concepts Lung-Infection-Database and Cancer-Database can be considered to be *intensionally related*, hence the following interschema semantic knowledge is asserted.

```
(Has-Gen-Spec-Conflict-With[7] Lung-Infection-Database  Cancer-Database).
(Intensionally-Related-With  Lung-Infection-Database  Cancer-Database).
```

[5] Recall in sub-section 2.1 that Pneumonia was defined in the MGKM as an infection located in the lung.

[6] *Abstraction level incompatibility conflict*: Refers to generalisation and aggregation conflicts. Two database concepts having this conflict are considered to be either intensionally or extensionally related

[7] Has-Gen-Spec-Conflict-With is a shorthand for Has-Generalisation-Specialisation-Conflict-With relation.

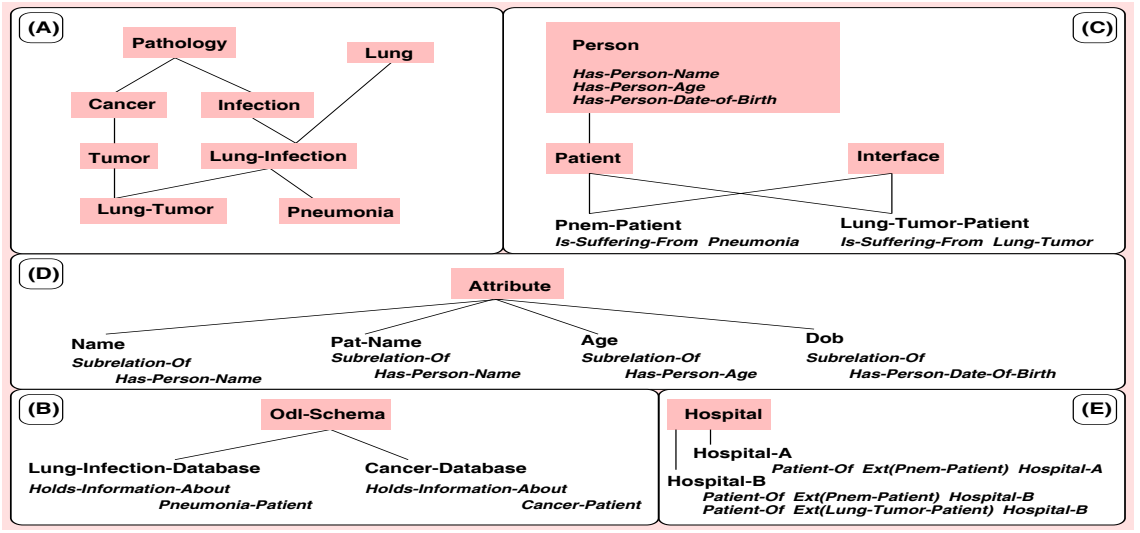


Fig. 3. Descriptive Meta-Knowledge of the Enriched Schema Models

This is corroborating evidence, which triggers the next phase of the identification process. However, the detected semantic knowledge at this coarsest level of schematic granularity is not yet sufficient to assign any value of semantic similarity about the domain values of the database schemas to be integrated.

The second phase of the identification process happens at the *schema level*. This phase builds on the knowledge gathered in the first phase (at the database level). Accordingly, the information acquired by the system at the database level triggers the identification phase at this level. In our simplified example the system attempts to check for a common or shared conceptualisation between the descriptions of the schema concepts Pnem-Patient and Lung-Tumor-Patient. Both schema concepts appear to be semantically equivalent at that level since they are both defined as instances of the Patient concept, see Figure 3 (C). This conclusion must be more fully examined by checking the rest of the semantic relationships associated with each schema concept. Here, the analysis of the Is-Suffering-From relationship refutes the possibility that both schema concepts are semantically equivalent. The Is-Suffering-From relationship is constrained to patients treated from Pneumonia in the Pnem-Patient schema, and from Lung-Tumor in the Lung-Tumor-Patient schema. The system also detects that Pneumonia and Lung-Tumor are both infections which are located in the Lung, (Lung-Infection, Figure 3 (A)). Hence, an *abstraction level incompatibility conflict* is identified and both schema concepts Pnem-Patient and Lung-Tumor-Patient can be considered to be *intensionally related* at this level, with the following interschema semantic knowledge asserted.

```
(Intensionally-Related-With Pnem-Patient Lung-Tumor-Patient).
```

Nevertheless, the acquired knowledge directs the system to trigger the next phase of the identification process, to be executed at the *attribute level*. The process of conflict detection at the *attribute* and *instance* levels follows a pattern similar to that associated with the *database* and *schema* levels: exploration, term expansion (specialisa-

tion/generalisation), conflict detection and interschema knowledge assertion. Similarly, the corroboration process gathers knowledge through the invocation of reconciliation techniques and the exploration of any knowledge or metadata that has previously been acquired by navigating the higher (i.e. coarser) levels of schematic granularity. At the attribute level we will only consider the description of the concepts representing the attributes Name and Age of the Lung-Infection-Database and the Pat-Name and Dob of the Cancer-Database.

The attributes Name and Pat-Name are both defined as *subrelations of* the Has-Person-Name relationship of the Person concept in the MGALEN knowledge model, see Figure 3 (D). Consequently, they are considered to be *intensionally equivalent* concepts having a *naming conflict*[8]. In addition, Pat-Name is the primary key of the Lung-Tumor-Patient schema, while on the other hand, Id is the primary key of the Pnem-Patient schema. The following interschema semantic knowledge is asserted.

```
(Is-Synonym-With Name Pat-Name).
(Intensionally-Equivalent Name Pat-Name).
(Has-Key-Equivalence-Conflict-With Pnem-Patient Lung-Tumor-Patient).
```

The concept representing the Age attribute of the Pnem-Patient schema is defined as a subrelation of the Has-Person-Age relation, and the concept representing the Dob attribute of the Lung-Tumor-Patient schema is defined as a subrelation of the Has-Person-Date-Of-Birth. Both of these relations have Person as their domain and Date as their range. It is also possible to determine the age of a person based on his date of birth. Associated with the Has-Person-Date-Of-Birth relation is the following *conversion function* which calculates the age of a person accordingly.

```
(Deffunction Date-Of-Birth-To-Age Person Dob Age)
    (=> (and (Person ?person)
             (Has-Person-Date-Of-Birth ?person ?dateOfBirth)
             (Year-Of ?dateOfBirth ?yearOfBirth)
             (Calendar-Date ?currentDate)
             (Year-Of ?currentDate ?currentYear)
             (= ?age (- ?currentYear ?yearOfBirth)))
        (Has-Person-Age ?person ?age)).
```

Hence a *data representation conflict* is detected[9]. This conflict may be resolved using the Date-Of-Birth-To-Age conversion function. Thus the following knowledge is asserted:

```
(Has-Data-Representation-Conflict-With Dob Age Date-Of-Birth-To-Age).
```

Since many ISs hold closely related information, the description of each schema concept should be able to model fine-grained differences between their contents, so that the interschema semantic knowledge may be accurately detected. In our example the Age concept of the Pnem-Patient schema is constrained to patients whose ages are less than or equal to 18 years old. On the other hand, there is no restriction on the Dob values recorded in the Lung-Tumor-Patient schema. Hence, the following knowledge is asserted:

[8] This conflict includes *synonyms* (where the same concept is described by two different names) and *homonyms* (in which the same name is used for representing two different concepts). Alternatively, a conflict can be classified as *unrelated* if the corresponding elements can not be categorised as either synonyms or homonyms.

[9] An object that can be mapped to another through a computed or a derived function.

```
(Subset Age Dob).
(Intensionally-Related-With Dob Age).
```

This assertion states that the attribute Age of the Pnem-Patient schema is intensionally less general than the Dob attribute of the Lung-Tumor-Patient schema. This means that the Dob concept *contains* the Age concept, and this containment is conceptual, not necessarily being reflected at the instance level. The process of conflict detection for the rest of the attributes of both schemas follows a similar pattern.

Concepts that are semantically related at the intensional level are not necessarily related at the extensional (instance) level. However, only when the corresponding concepts have some intensional relationship may they be extensionally related. In our example, the intensional knowledge gathered during the three previous phases triggers the identification phase at the instance level. Based on the knowledge acquired during the enrichment process, the set of instances (extension) of the concept representing the Pnem-Patient table intersects with those represented by the Lung-Tumor-Patient concept of the Lung-Tumor-Patient class, see Figure 3 (E). Accordingly, both schema concepts can be considered to be *extensionally related*, hence the following interschema semantic knowledge is asserted:

```
(and (Pnem-Patient ?Pat1)
     (Extent ?Pat1 ?Pat1-Extent)
     (Lung-Tumor-Patient ?Pat2)
     (Extent ?Pat2 ?Pat2-Extent)
     (Intersect-With ?Pat1-Extent ?Pat2-Extent)).
```

Based on the above asserted fact, and the interschema knowledge acquired at the *database, schema, and attribute levels* (that both schema concepts are intensionally related) the following interschema knowledge is asserted:

```
(Extensionally-Related-With Pnem-Patient Lung-Tumor-Patient).
```

The process of schema analysis and interschema correspondence assertion between classes belonging to different schemas continues until reconciliation is achieved (e.g. all corresponding schema concepts have been mapped with confirmed assertions), or no further discovery of interschema knowledge is possible. In both cases human intervention is required to confirm or reject newly determined interschema knowledge or the termination of the identification process.

5 Conclusions

We have presented a formal approach for discovering and representing interschema knowledge in a cooperative MIS environment. Interschema semantic knowledge is specified in terms of a semantically rich knowledge model (MGKM), in an attempt to make explicit the knowledge which a human integrator uses implicitly to identify semantically similar schema concepts. The MGKM is an integral part of the MetaMed integration system which allows us to create semantically rich descriptions of the ISs by *lifting* their structural and semantic content into the MGKM vocabulary [22, 23], and for expressing several forms of semantic interdependencies between the classes represented in different schemas. The main contribution of our approach is that we express semantic interdependencies between the descriptions of the classes belonging to different IS

systems rather than a simple mapping of those classes to the semantic model. Hence, our approach provides more extensive semantic knowledge about these classes, which proved to be necessary in discovering and reusing interschema knowledge when the need arises to compare and combine it. The framework presented in this paper concentrates on expressing conceptual and semantic correspondences between both the conceptual (intensional) definition and the set of instances (extension) of classes represented in different schemas.

The adoption of a declarative encoding of data semantics brings about many benefits, chief among which is the ability to make explicit the interschema semantic knowledge and the ability to query directly the semantics of the data which are implicit in different IS systems [21]. At the same time, the semantic nature of our approach makes it clear that there are several issues which warrant further investigation, among those are issues related to interschema knowledge representation and reasoning, such as: expressivity, complexity and tractability.

References

1. R. Ahmed, P. De Smedt, W. Du, W. Kent, M. A. Ketabchi, W. Litwin, and A. Rafi. The Pegasus heterogeneous multidatabase system. *IEEE Computer*, 24(12):19–27, 1991.
2. Y. Arens, C. Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, November 1986.
4. E. Bertino. Integration of heterogeneous data repositories by using object-oriented views. *Proceedings of International Workshop on Interoperability in Multidatabase Systems, Kyoto, Japan*, pages 22–29, 1991.
5. P. Buneman, L. Raschid, and J. Ullman. Mediator languages - a proposal for a standard. Report of an I3/POB Working Group held at the University of Maryland, ftp://ftp.umiacs.umd.edu/pub/ONRrept/medmodel96.ps, April 1996.
6. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
7. R.G. Cattel and Morgan Kufmann. *ODMG-93 The Object Database Standard Release 1.2*. Inc., San Francisco, California, 1996.
8. C. Collet, M. N. Huhus, and W. M. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, 1991.
9. C. H. Goh, S. Bressan, S. Mandick, and M. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–293, July 1999.
10. D. D. Karunaratna, W. A. Gray, and N. J. Fiddian. Establishing a knowledge base to assist integration of heterogeneous databases. In Suzanne M. Embury et al, editor, *Proceedings of 16th British National Conference on Databases (BNCOD16)*, LNCS 1405, Springer, pages 103–118, 1998.
11. M. Kifer and G. Lauson. F-logic: A higher-order language for reasoning about objects, inheritance and scheme. *Proceedings of ACM SIGMOD Conference*, 1995.

12. R. Krishnamurthy, W. Litwin, and W. Kent. Language features for interoperability of databases with schematic discrepancies. *Proceedings of ACM SIGMOD Conference*, 1991.
13. E. Kuhn and T. Ludwig. VIP-MDBMS: A logic multidatabase system. *Proceedings of 2nd International Symposium on Distributed Databases*, 1988.
14. J. Larson, S. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.
15. Alon Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2), 1995.
16. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Proceedings of First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Brussels (Belgium), IEEE Computer Society Press*, pages 14–25, June 1996.
17. A. Motro. Superviews: Virtual integration of multiple databases. *IEEE Transactions on Software Engineering*, 13(7):785–798, 1987.
18. A. M. Ouksel and I. Ahmed. Ontologies are not the panacea in data integration: A flexible coordinator to mediate context construction. *Distributed and Parallel Databases, Kluwer Academic Publishers*, 7(1):7–35, 1999.
19. A. Rector, W. Solomon, W. Nowlan, and T. Rush. A terminology server for medical language and medical information systems. *Methods of Information in Medicine*, 34:147–157, 1995.
20. M. Siegel and S. Madnick. A metadata approach to resolving semantic conflicts. *Proceedings of 17th VLDB Conference*, 1991.
21. A-R Tawil and Wernher Behrendt. Requirements for components of an intelligent information retrieval model for the www. *Intelligent World Wide Web Agents. IEE Colloquium organised by Professional Group c4 (Artificial Intelligence)*, 97(118):1/1 – 1/7, March 1997.
22. A-R Tawil, W. A. Gray, and N. J. Fiddian. Ontological commitments for multiple view cooperation in a distributed heterogeneous environment. *Proceedings of 16th British National Conference on Databases (BNCOD16)*, 1998.
23. A-R Tawil, W. A. Gray, and N. J. Fiddian. Rich semantic-based schema integration in a federated MDBS environment. *Submitted for publication to IDEAS*, 2000.
24. William A. Woods and James G. Schmolze. The KL-ONE family. *Computer Math. Applic.*, 23(2-5):133–177, August 1992.

A Description Logics-Like Model for a Knowledge and Data Management System

Mathieu ROGER, Ana SIMONET, Michel SIMONET

TIMC-IMAG Faculté de Médecine de Grenoble
38706 La Tronche Cedex – France
Mathieu.Roger@imag.fr, {Ana,Michel}.Simonet@imag.fr

Abstract. *Nowadays data management systems and knowledge management systems tend to converge. Our work is part of the OSIRIS data and knowledge management system. This system is centered on views which are said object-preserving (i.e. they do not create new objects). Our work led to the construction of a model for the OSIRIS System inspired from those of Description Logics. In this model we have defined the main functionalities of databases (insert, delete, modify, select) and description logics (subsumption). In this manner we show that there is a close relationship between subsumption calculus and query evaluation with the notion of interpretation. The data stored in a database can be seen as an interpretation (same meaning as in first order logic) of the set of axioms constituted by the databases' scheme. Typically a database manipulates what is true in a single interpretation, a knowledge base what is true in all possible interpretations.*

1 Introduction

Databases and knowledge bases constitute two distinct domains of computer science. They originated separately and have developed independently. However, it now appears that their needs are not specific to each domain. Merging databases with knowledge bases is not a new idea, and some systems already implement such a blend: for example Datalog (merging of database and Prolog), the use of rules for checking integrity constraints [Roncancio, 94], ConceptBase and the Osiris system [Simonet et al., 94].

Knowledge bases typically rely on various formalisms, such as rule-based systems, conceptual graphs or description logics. The latter are part of the more general paradigm of model theory. In fact, they consist of several concept constructors and these concepts can be seen as classes. Description Logics are interesting in two ways. First they permit to define a deduction operator between concepts, namely subsumption. Second they are close to the object-oriented paradigm because primitive concepts can be seen as classes and defined concepts as views. Therefore, Description Logics seem well fitted to the merging of object-oriented and deductive approaches [Buchheit et al., 94].

We focus on the fact that classical approaches to databases and description logics have common points but do not operate on the same level of data. In effect, description logics are concerned by the scheme in intension (i.e. whatever the

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 563-572, 2000.

interpretation is) whereas databases are concerned by the scheme in extension (i.e. for a particular interpretation). The work in [Domini et al., 96] also uses this idea.
The goal of this paper is double. First, it seemed necessary to restrict the language of description logics to match the specific needs of databases. Then we show a way to unify database selection queries and subsumption.
This paper is organised as follows: in section 2 we briefly describe the OSIRIS system. Section 3 and 4 define the OSIRIS specification language, first syntactically then semantically. Specific applications are defined in section 5. Finally we outline future work.

2 Main Characteristics of the Osiris system

The Osiris system is an implementation of the p-type model defined in [Sales-Simonet, 84]. This model aims at sharing data through views and statically analysing integrity constraints. In Osiris views are object-preserving [Scholl et al., 91], i.e., they are defined as subsets of existing objects. Object-generating views (i.e., views that create new objects, as in relational systems) are not permitted. This functionality will be implemented by allowing the creation of new types whose extensions will be defined as general queries. A view is defined as a set of attributes and logical formulas whose elementary predicates are of the type *attribute* $\in$ *domain*.
An object belongs to a view's extension iff it satisfies all formulas of the views. For example, if we define the view *Minor* as *Persons* less than 18 years old, then the system deduces that a *Person* x whose age is 24 does not belong to Minor's extension.
The system indexes objects according to elementary predicates used in the views' definitions by partitioning the object space (defined by all the possible attribute assignations for an object). This is one of the main features of Osiris. It allows us to have updatable views, i.e. views through which users can interact and for example insert objects, according to [ANSI/X3].

3 Scheme Language Syntax

In this section we give a definition of the concept language syntax for OSIRIS. On an operational point of view, this language is used to specify schemes.
Let us consider a set of types *T* (ranged over by A,B,T) containing predefined types INT, REAL, CHAR, STRING, namely Predefined, a set of view *V* (ranged over by U,V) and a set of roles *R* (ranged over by R). These are sets of distinct names.
As in [Buchheit et al., 98] we use two distinct languages: one for the types (corresponding to primitive concepts), and another one for the views (corresponding to defined concepts).

Definition 1. Given T,V and R we inductively define the *type concept language* (namely TL) and the function *attribute*: TL$\rightarrow$ P(R$\times$T):

- $\forall$ R.A, where R$\in$ *R* and A$\in$ *T*, and attribute($\forall$ R.A) = {(R,A)}

- $\forall$ R.A $\cap$ (= 1 R), where R$\in$ *R* and A$\in$ *T*, and attribute($\forall$ R.A$\cap$ (= 1 R)) = {(R,A)}
- C $\cap$ D, where C and D are type concept, and attribute(C $\cap$ D) = attribute(C) $\cup$ attribute(D)

Definition 2. Given *T*,*V* and *R* we inductively define the view concept language (namely VL):

- A, where A$\in$ *T*
- V, where V $\in$ *V*
- U $\cap$ V, where U and V are view concepts
- $\neg$Undefined(R), where R$\in$ *R*
- $\forall$ R.]v_1,v_2[, $\forall$ R. [v_1,v_2[, $\forall$ R.]v_1,v_2] , $\forall$ R. [v_1,v_2] where v_1 and v_2 are both elements of one of the types INT, CHAR or REAL
- $\forall$ R.{a_1,…,a_n}, where a_i are all elements of one of the types INT,REAL, CHAR or STRING

The two latter cases are called *elementary constraints*. They are of the type attribute $\in$ domain.

Definition 3. Given *T*,*V* and *R* a type scheme is a set S of axioms of the type:

1. A $\subseteq$ C, where A $\in$ *T* and C $\in$ TL
2. A $\subseteq$ $\neg$B, where A $\in$ *T* and B$\in$ *T*
3. R $\subseteq$ A$\times$B, where R$\in$ *R*, A$\in$ *T* and B $\in$ *T*
4. *V* = *U, where* V $\in$ *V* and U $\in$ VL

Such that:

- *Uniqueness of a type definition:* for all A $\in$ *T*, there is one axiom A $\subseteq$ C in S. Such types are called *p-types*.
- *A role belongs to a single type and all roles are defined:* for all R there is exactly one axiom R $\subseteq$ A$\times$B and A $\subseteq$ C such as C uses R.
- *Types are disjoint:* for all A,B in *T*, there is an axiom A $\subseteq$ $\neg$B.
- *Uniqueness of a view definition:* for all V$\in$ *V* there is exactly one axiom V = U.
- *Views form a hierarchy:* considering the binary relation of view inclusion "V uses U in its definition, V = U$\cap$ …", the directed graph formed by views as node and this relation is acyclic and every connex compound has a unique root called *minimal view*.
- *A view is a subset of a unique type:* for each view V there is exactly one type A such as minimal_view(V) = A $\cap$ ….

Informally these axioms have the following meaning:

1. Types are given by a set of roles
2. Types are disjoints
3. Roles have a domain and a codomain
4. Views are definitions

Remarks. We have decided to separate type definitions from view definitions as in [Buchheit et al., 98] for two reasons. First, this follows the architecture advocated by [ANSI/X3]: views are external schemes and types form the global internal scheme. In the Osiris system types are built from views' definitions. Second, distinguishing views from types focuses on the fact that these two concepts have not the same usage. In effect, a user must explicitly state that an object belongs to a primitive concept: the system cannot deduce such a belonging. So, a primitive concept is used in order to model a partially described set of objects, whereas a defined concept is completely characterised (relatively to primitive concepts) [Doyle et Patil, 91]. In example 1, PERSON is a primitive concept: if an object has an *integer* age, a *string* name and partners who are PERSON it is not necessarily a PERSON. In other words, types defined in this paper should be seen as types in many programming languages, which are in fact primitive concepts. The language Ada allows constraints on types such as *integer in [2..10]*, this kind of constraints is an analogous to our views in a programming language.

Definition 4. Given S a type scheme and A in *T*-Predefined such that $A \subseteq C$ is in S we define attribute(A) as attribute(C).

Definition 5. Given S a type scheme we define the function type from *V* to *T* that associates a view to its type. Given S a type scheme we define function roles from *T* to P(*R*), which associates a type to roles used in its description.

Property. Given S a type scheme, then for all A in *T*, attribute(A) is the graph of a total function from $R \rightarrow T$. Thus we could write attribute(A)(R) for the type of attribute R from type A.

Example 1. Given *T*={INT, STRING, PERSON, COURSE}, *R*={partner, age, namePERSON, nameCOURSE, follow, teach, teacher}, *V*={VIEWPERSON, VIEWCOURSE, STUDENT, TEACHER, TEACHINGASSISTANT} we can define the following type scheme.
Following are constraints for types:

- PERSON $\subseteq$ $\forall$partners. PERSON $\cap$ $\forall$ age. INT $\cap$ (= 1 age) $\cap$ $\forall$follow. COURSE $\cap$ $\forall$teach. COURSE $\cap$ $\forall$ namePERSON. STRING $\cap$ (= 1 namePERSON)
- COURSE $\subseteq$ $\forall$teacher. PERSON $\cap$ (= 1 teacher) $\cap$ $\forall$ nameCOURSE. STRING$\cap$ (= 1 nameCOURSE)

These are constraints telling that types are disjoint:

- PERSON $\subseteq \neg$ COURSE
- PERSON $\subseteq \neg$INT
- INT $\subseteq \neg$ COURSE
- INT $\subseteq \neg$STRING
- PERSON $\subseteq \neg$STRING
- COURSE $\subseteq \neg$STRING

These are constraints for view definitions:

- VIEWPERSON = PERSON ∩ ¬Undefined(age) ∩ ¬Undefined(partner) ∩ ¬Undefined(namePERSON) ∩ ∀ age. [0, 140]
- STUDENT = VIEWPERSON ∩ ¬ Undefined (follow)
- TEACHER = VIEWPERSON ∩ ¬ Undefined (teach)
- TEACHINGASSISTANT = STUDENT ∩ TEACHER
- MAJOR = VIEWPERSON ∩ ∀ age. [18, +∞[
- VIEWCOURSE = COURSE ∩ ¬ Undefined(teacher) ∩ ¬ Undefined(nameCOURSE)

This is a description logics-like notation but we prefer a notation like classical databases. The following example is the same as example 1 in the practical Osiris description language.

Example 2

```
p-type PERSON
view VIEWPERSON
attributes
          name : STRING ;
          age : INT ;
          partners : set of PERSON;
assertions
          age in [0,140] ;
end;
view STUDENT : VIEWPERSON
attributes
          follow : set of COURSE;
end;
view TEACHER : VIEWPERSON
attributes
          teach : set of COURSE ;
end ;
view TEACHINGASSISTANT :STUDENT, TEACHER
end;
view MAJOR : VIEWPERSON
assertions
          age >= 18 ;
end;
p-type COURSE
view VIEWCOURSE
attributes
          teacher : PERSON;
          name : STRING ;
end;
end;
```

4. Semantics

Definition 6. An interpretation $I = (\Delta^I, .^I)$ of a given type scheme S is given by a set Δ^I and an interpretation function such that starting concepts and roles are interpreted in the following way:

- A^I is a subset of Δ^I for all A in *T*
- V^I is a subset of Δ^I for all V in *V*
- R^I is a triplet <Relation, Undefined, Unknown> with Relation $\subseteq \Delta^I \times \Delta^I$, Undefined $\subseteq \Delta^I$, Unknown $\subseteq \Delta^I$ and $\{x \in \Delta^I \mid \exists\ y \in \Delta^I$ and $(x,y) \in$ Relation$\} \subseteq \Delta^I$ - (Undefined $\cup$ Unknown)

And such as complex concepts are interpreted according to the following rules:

- $(\neg B)^I = \{x \in \Delta^I \mid x \notin B^I\}$, where $B \in T$
- $(\forall R.A)^I = \{x \in \Delta^I \mid \forall y \in \Delta^I: (x,y) \in Rel^I \Rightarrow y \in A^I\}$, where $A \in T$
- $(C \cap D)^I = C^I \cap D^I$, where C and D are view concepts or C and D are in TL
- $((= 1\ R))^I = \{x \in \Delta^I \mid \#\{y \in \Delta^I: (x,y) \in R^I\} = 1\ \}$, where $R \in R$
- $(\neg Undefined(R))^I = \{x \in \Delta^I \mid x \notin$ Undefined$\}$, where $R^I =$ <Relation, Undefined, Unknown>

Remarks. The undefined set in a role interpretation represents the objects for which the role has no meaningful value. It is the case for the function 1/x: it is undefined for x=0. As in Description Logics we assume that names are unique (unique name assumption), in the context of our database system these names are oids.

Definition 7. Given I and J two interpretations of a scheme S, we say that J is a more specific interpretation than I (noted as $J \subseteq I$) iff:

- $\Delta^I \subseteq \Delta^J$
- for all A in *T*, $A^I \subseteq A^J$.
- for all V in *V*, $V^I \subseteq V^J$.
- for all R in R, $R^I =$ <$Relation_1$, $Undefined_1$, $Unknown_1$>, $R^J =$ <$Relation_2$, $Undefined_2$, $Unknown_2$>, $Relation_1 \subseteq Relation_2$, $Undefined_1 \subseteq Undefined_2$ and $Unknown_2 \subseteq Unknown_1$.

In other words, J a more specific interpretation than I contains less unknown information and possibly more objects than I.

Definition 8. An interpretation I is said finite if Δ^I is finite.

Definition 9. Given a scheme S, we say that a finite interpretation I may satisfies an axiom. More precisely:

1. I satisfies $A \subseteq C$ iff $A^I \subseteq C^I$
2. I satisfies $A \subseteq \neg B$ iff $A^I \cap B^I = \varnothing$

3. I satisfies $R \subseteq A \times B$ iff Relation $\subseteq A^I \times B^I$, Undefined $\subseteq A^I$ and Unknown $\subseteq A^I$, where $R^I = <$ Relation, Undefined, Unknown$>$
4. I satisfies $V = U$ iff $V^I = U^I$

We say that a finite interpretation *satisfies* a type scheme S (or that I is a valid interpretation of S) iff:

- I satisfies every axiom of S
- There exist $J \subseteq I$ such that J satisfies every axiom of S and Unknown$(R^J) = \varnothing$.

Definition 10. We say that a view V_1 subsumes a view V_2, iff $V_2^I \subseteq V_1^I$ for all valid interpretation I.

5. Calculus of Subsumption

The goal of this section is to show how subsumption can be calculated by defining a space where views are subsets and subsumption is set inclusion.

Definition 11. Given a type scheme S, a view V and I a valid interpretation of S, we say that the virtual space of V with regard to I is

$$\text{Virtual}(V,I) = \times_{\text{for all r in role(type(V))}} (\{y \mid \exists x \in V^I, <x,y> \in \text{Relation}\} \cup \{\text{undefined} \mid \exists x \in V^I, x \in \text{Undefined}) \text{ where } r^I = <\text{Relation,Undefined ,Unknown} >$$

Definition 12. The virtual space of a view V, namely virtual(V), is the union of virtual(V,I) for all valid interpretation I.

Definition 13. The virtual space of a type A is the set:

$$\text{Virtual}(A) = \times_{\text{for all r in role(A)}} (B \cup \{\text{undefined}\}) \text{ where } r \subseteq A \times B$$

We will now show that virtual spaces of views can be finitely represented by eq-classes. This is a consequence of the choice made for the elementary constraints.

Definition 14. Given a type scheme S and one of its role R, $R \subseteq A \times B$ is an axiom of S, the domain of R, namely domain(R), is:

- Set(B), i.e. the set of all elements of type B, if R considered as an attribute is single-valued
- Set(set(B)) otherwise

Definition 15. The stable sub-domains of a role R, namely SSD(R), are subsets of domain(R) such as:

1. SSD(R) is a partition of domain(R)

2. For each elementary constraint upon R, $\forall$ R.C (where C is]a,b], …,or {a, b, c}) there are ssd_1, …, $ssd_n \in$ SSD(R) such as C = $ssd_1 \cup \ldots \cup ssd_n$.
3. The number of stable sub-domains is minimal (condition 1 and 2 does not provide a unique set of SSD)
4. {undefined } $\in$ SSD(R)

All the elementary constraints can be written as $attribute \in ssd_1 \cup \ldots \cup ssd_n$. Logic constraints for view satisfaction are thus predicate calculus formulas [Simonet et al., 94].

Example 3. Stable sub-domains for attributes in example 2 are:

- SSD(age) = {[MININT, 0[, [0,18[, [18,140],]140,MAXINT], {undefined}}
- SSD(name) = {STRING, {undefined}}
- SSD(partners) = { set(PERSON), {undefined}}
- SSD(follow) = {set(COURSE), {undefined}}
- SSD(teach) = {set(COURSE), { undefined}}

Definition 16. Let A be a p-type, an eq-class of A is a set:

$$\times_{\text{for all r in role(A)}} \text{ssd(r)}, \text{ where ssd(r)} \in \text{SSD(r)}$$

In fact, eq-classes of A form a partition of virtual(A).

Example 4. Example of eq-class
([0,18[, STRING, set(PERSON), set(COURSE), {undefined})

Property. The virtual space of a view is the union of a finite number of eq-classes.

Proof. c.f. [Sales-Simonet 84]

Property. A view V_1 subsumes a view V_2, iff virtual(V_1) $\subseteq$ virtual(V_2).

Proof. By induction over the number and kind of constraints. c.f. [Sales-Simonet 84].

An object can be seen as an element of the virtual space, or a subset if the object is not completely known, by projecting its attributes' values in this space. A formula such as PERSON $\cap$ $\forall$ age.[15,40] can be seen, in intension, as a subset of the virtual space too and, in extension, as the set of known objects matching the query.

6. Conclusion and Future Work

The semantics given to Description Logics are formal ones contrary to most database standards the models of which are often expressed in natural language (however, we

note the work of [Coupaye, 96]). So we have tried to clearly specify that the use of inheritance in the Osiris system is "set inclusion". The use of a Description Logics model appeared to us as a good candidate to perform such a clarification.
Furthermore, the vision of both objects and logical formulas as subsets in the virtual space has led good results in the semantic object indexation in Osiris. We hope that this will set a bridge between database and knowledge base domains. It also seems to be interesting for semantic query optimization as a query can be evaluated intensionally first, then extensionally.
The objectives of our work are double. First one is to increase the expression power of the views by adding new types of constraints such as linear constraints or constraints of the form *attribute* $\in$ *view*.
Second one is to add other notions of inheritance to this model. This is a necessary step toward a full object system and should be closely linked with view inheritance, thus providing an intelligent data environment.

References

[Abiteboul et al., 89]: S. Abiteboul, P. Kanellakis, *Object identity as a query language primitive*, SIGMOD Record (ACM Special Interest Group on Management of Data), n° 18(2), pp. 159-173, Juin 1989.

[AFIA, 98]: Bulletin de l'AFIA n° 34, juillet 1998, Dossier Ingénierie des Connaissances, pp 65-66.

[Alsys, 87]: Alsys, *Manuel de référence du langage de programmation Ada*, 1987.

[ANSI/X3]: ANSI/ X3/ SPARC Study group on database management systems. Interim report, ACM SiGMOD Bulletin 7, N2, 1975.

[Beneventano et al., 93]: Beneventano, Bergamaschi, Lodi, Sartori, *Using subsumption in semantic query optimisation*, IJCAI Workshop on object based representation systems, Août 1993.

[Bert et al., 95]: D. Bert, R. Echahed, P. Jacquet, ML. Potet et JC. Reynaud, *Spécification, Généricité, Prototypage: Aspects du langage LPG*, Journal of Technique et Science Informatiques 9 vol 14. pp 1097-1129. 1995.

[Bertino, 92]: E. Bertino, *A View Mechanism for Object-Oriented Databases*, Advances in DB-Technology, Proc. Intl. Conf. on Extending Database Technology (EDBT), Lecture Notes in Computer Science, Number 580, pp. 136-151, Springer, March 1992.

[Bertino et al., 92]: E. Bertino and G. Guerrini, *Objects with Multiple Most Specific Classes*, ECOOP'95---Object-Oriented Programming, 9th European Conference, Lecture Notes in Computer Science, Vol. 952, pp. 102-126, Springer, 7-11 August 1995.

[Bucheit et al., 94]: M. Buchheit, M. Jeusfeld, M. Staud, W. Nutt, *Subsumption between Queries to Object-Oriented Databases*, EDBT, 1994.

[Bucheit et al., 98]: Buchheit, Domini, Nutt, Schaerf, *A refined architecture for terminological systems : terminology = schema + views*, Artificial Intelligence, Vol 99, 1998.

[CLASSIC]: *CLASSIC Description and Reference Manual For the COMMON LISP Implementation Version 2.2*.

[Coupaye, 96]: T. Coupaye, *Un modèle d'exécution paramétrique pour systèmes de bases de données*, Thèse, LSR-IMAG Université Joseph Fourier, 1996.

[Domini et al., 96]: Domini, Lenzerini, Nardi, Schaerf, *Reasoning in description logics*, Principles of Knowledge Representation, pp. 191-236, CSLI Publications, 1996.

[Doyle et Patil, 91]: J. Doyle, R. Patil, *Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation srvices*, in Journal of Artificial Intelligence, vol 48, 1991

[Euzenat, 98]: J. Euzenat, *COURSE de sémantique des modèles de connaissances*, DEA d'informatique, Grenoble, 1998.

[Hollunder]: B. Hollunder, Werner Nutt, *Subsumption algorithms for concept languages*, Research Report, Deutsches Forschungszentrum für Künstliche Intelligenz, Number RR-90-04.

[Kim et al., 95]: W. Kim (ed), *Modern Database Systems. The Object Model Interoperability and Beyond*, ACM Press, Addison Wesey Publishing Company, 1995.

[Lacroix et al., 98]: Z. Lacroix and C. Delobel and P. Breche, *Object Views and Database Restructuring*, Lecture Notes in Computer Science, Vol. 1369, 1998.

[Napoli]: A. Napoli, *Une introduction aux logiques de description*, Technical Report, Inria, Institut National de Recherche en Informatique et en Automatique, Number RR-3314.

[Nebel, 90]: B. Nebel, *Reasoning and Revision in hybrid representation systems*, LNAI, n°422, 1990.

[Sales-Simonet, 84]: A. Sales-Simonet, *Types abstraits et bases de données*, Thèse, Université scientifique et médicale de Grenoble,1984.

[Scholl et al., 91]: Scholl, Laasch, Tresch, *Updatable Views in Object-Oriented Databases*, Proc. 2nd Intl. Conf. on Deductive and Object-Oriented Databases (DOOD), Lecture Notes in Computer Science, n° 566, 1991.

[Simonet et al., 94]: A. Simonet, M. Simonet, *Objects with Views and Constraints : from Databases to Knowledge Bases*, Object-Oriented Information Systems OOIS'94 - London, Springer Verlag, pp 182-197, Dec. 1994.

[Simonet, 88]: A. Simonet, *Les P-TYPES: un modèle pour la définition de bases de connaissances centrées-objets cohérentes*, R.R. 751-I laboratoire Artemis, Grenoble, Novembre 1988.

[Simonet et al., 98]: A. Simonet, M. Simonet, C. G. Bassolet, X. Delannoy, R. Hamadi, *Static Classification Schemes for an Object System*, In: FLAIRS-98, 11th Int. FLorida Artificial Intelligence Research Society Conference, AAAI Press, pp 254-258, May 1998.

[Souza dos Santos et al., 94]: Souza dos Santos, Abiteboul, Delobel, Virtual schemas and bases, LNCS 779, 1994.

[Roger, 99]: M. Roger, *Requêtes dans un SGBD-BC de type objet avec vues*, Rapport de Dea, UFR IMA, Grenoble, 1999.

[Roncancio, 94]: C. Roncancio, *Règles actives et règles déductives dans les bases de données à objets*, Thèse, Université Joseph Fourier, 1994.

An Ontology Driven Approach to Ontology Translation

Houria Mihoubi[1] and Ana Simonet[1] and Michel Simonet[1]

Laboratoire TIMC-IMAG, Faculté de Médecine de Grenoble
38706 La Tronche Cedex - France
E-mail : {houria mihoubi, ana simonet, michel simonet}@imag.fr

Abstract. Translating ontologies from a source language towards a target one is a required process both for the conception of a new ontology from existing ones (reuse) and for common use of an ontology by different knowledge based systems. The usual approach for translation is based upon the existence of a translator for each pair of languages from and to which the ontology must be translated. Therefore, several translating tools are necessary.
We propose a translation approach which employs a unique translator no matter what source and target languages are involved. It relies on two principal components. The first one is a meta-language called `M_Kif` which extends the `Kif` language with the concept of meta-relation to describe representation ontologies. The second one is a pivot representation which unifies different styles of representations. The translation tool is an interpretation program of the meta-relation definitions specified in the source and target representation ontologies.

1 Introduction

During the last decade, the computer science community and particularly the artificial intelligence community has shown a growing interest in work on ontologies. Among these works the ontology design and the ontology use by knowledge based systems (KBS) are the subject of considerable research activity. The aim is to provide the means, formalisms and tools which contribute to the use of ontologies by heterogeneous systems, by facilitating the exchange of knowledge between them, and to the design of new ontologies from existing ones, i.e., the **reuse**.

If ontologies are widely acknowledged as an efficient means of knowledge sharing and reuse, their design is, on the other hand, a long, costly and difficult process. There are two main reasons for this. The first one is the lack of tested methodologies of ontology design. Most existing ontologies have been developed to meet specific needs, and each team has followed its own methodology of which an overview is given in [2]. The second main reason is that those who develop ontologies are generally not experts in the domain characterised by the ontologies. A time consuming and costly process of knowledge acquisition is therefore necessary; ontology reuse may be a possibility of reducing conception costs.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 573–582, 2000.

When an ontology is constructed (whether or not by reuse) it may be used by different KBS, each providing different reasoning services. For example, if an application necessitates classifying concepts of an ontology and detecting inconsistencies, a knowledge representation system based on Description Logics (DLs) such as `Loom` or `Classic` may be required. The same ontology may also be used by a planning system if it is necessary to produce plans.

However, to achieve ontology use and reuse, it is necessary to supply ontology and KBS designers with technologies which are capable of supporting their activities. In these technologies, ontology translators are indispensable tools.

We propose a declarative approach for the automatic translation of ontologies which aims at using a single translator whatever the source and the target ontology language. This approach relies on a meta language, called `M_Kif` which enables us to describe languages based on relations, frames, DLs or objects, in a declarative manner. For a given language, this description constitutes its `Representation Ontology`. An other important component of our approach is a `Pivot Representation` which enables us to unify different knowledge representations. In our approach, the translation process of an ontology from a language L_1 to L_2 is entirely driven by the representation ontologies of L_1 and L_2.

To present this work, we begin, in section 2, by presenting the requirements for ontology reuse. In section 3 we motivate the need to ontology translation. To situate our approach, section 4 is devoted to the description of the usual translation tools. Our general approach is described in section 5. In sections 6, 7 and 8 we describe respectively the pivot representation, the `M_Kif` language and the translation process . Finally, in our conclusion we evaluate our work.

2 Reuse Requirements

New ontologies can be constructed by assembling existing ontologies. The latter can either be included as such or adapted, for example specialised. Sometimes, a new ontology is built entirely by reuse, as is the case of `PHYSYS`, a formal ontology based upon the system dynamics theory. However, ontology reuse is in itself a difficult and costly task which necessitates an adequate environment which allows reused ontologies to be integrated successfully. This environment must offer at least (1) the availability and the public access to the ontologies to be reused, (2) their documentation, (3) their translation, and (4) specific operators which make ontology assembly possible. The most efficient framework for obtaining information on existing ontologies, and which gives access to them is a public ontology server on the web. One of the most well known and widely used ontology servers is the Ontolingua server[1]. It's an ontology development environment which offers a library of public ontologies, facilities and operators which enable designers to edit, browse and reuse the library ontologies. Operators which make reuse possible are "inclusion" consisting in importing existing ontologies into a new one with no modification, "restriction" consisting in an inclusion followed by

[1] http://www-ksl-svc.stanford.edu:5915/

a restriction of the imported ontologies by adding constraints which limit their set of axioms, and "polymorphic refinement" which means that an operation can be extended for use with several types of arguments.

3 Why Translate Ontologies

The translation process of ontologies is necessary both for reuse and use by KBS. The ontologies to be reused must be translated in such a way they can be expressed in the same language as the new ontology. If an ontology O_1 described in a language L_1 must be reused by an ontology O described in a language L, with $L_1 \neq L$, then, it is necessary to translate O_1 from L_1 to L.

When used by a KBS, the ontology must be translated into the KBS representation language. Ontologies are constructed to be used in one or several applications which all have their own knowledge representation language, which may differ from the descriptive language of the ontology. For example, in the context of inter-operation between different applications, an ontology can be used as an interchange format. The translation process between the ontology and the application formats is required because each application needs to read and write the same data.

Translating an ontology means translating its generic terms, i.e., terms which have been used to describe the knowledge it carries. This set of generic terms is called **meta-ontology** in [9], **Frame Ontology** in [4], but the most commonly used name is **representation ontology**.

4 Usual Translation Approach

A classical translation tool works both at the symbolic and syntaxic levels. Using a certain number of pre-established rules, it follows two criteria.

1. it only applies to a pair of languages (L_1,L_2);
2. it is uni-directional: L_1 -> L_2 or L_2 -> L_1.

The translation process of the CORE model, a medical ontology developed in Grail, a knowledge representation system (KRS) based on DLs, into Loom - another KRS also based on DLs - consisted in two stages: analysis/pre-processing and code generation [5]. In the first stage, the Grail formulae were converted to Lisp. The result is then given to a translation module which generates the Loom code. Separating the process into two stages means that the analysis module could be reused for a translation from grail into other Lisp based systems such as Classic.

These are few experiments of reuse of translated ontologies: here we can note [10] where the EngMath ontology expressed in Ontolingua [4], was reused in a new specification expressed in Slang - a knowledge level specification language. Amongst the problems raised by this experiment, was the need to check the semantic differences between the set theory and the category theory on which Ontolingua and Slang are respectively founded, in order to detect possible inconsistencies.

To translate an ontology into N target languages, `N*(N-1)` translators are needed, as is illustrated in Fig. 1(a), in which each arrow represents one translator.

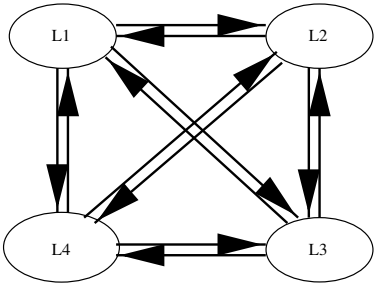


Fig. 1 (a)

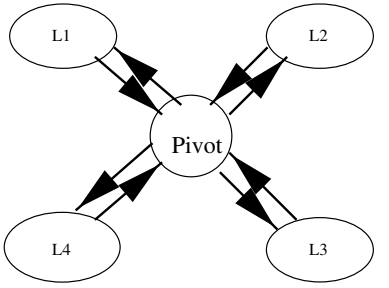


Fig. 1 (b)

The use of a pivot language (Fig. 1 (b)) such as `Kif` [3] considerably reduces the number of translators, which can be brought down to `(2 * N)`.

Certain ontology description languages are "naturally" accompanied by their translation tools. This is the case of the `Ontolingua` language [4], a structured form of `Kif`, which has its own translators. Ontolingua is a system for describing portable ontologies over the `Loom`, `Epikit` and `Algernon` systems. The `Enterprise` ontology [9] has been described using `Ontolingua`. Ontologies can be described using class, relation, function, and object definitions. These definitions can be written in `Kif` and/or by using terms specific to Ontolingua, called "second-order terms", defined in a representation ontology called the Frame Ontology. These terms correspond to a set of some common idioms that are supported in most target systems. Because `Kif` is very expressive, all of its sentences do not have an equivalent in the target systems. The `Kif` sentences which can be translated are written in the Frame Ontology. When Ontolingua finds these sentences, it transforms them into a canonical form before passing them on to the appropriate translator. In other words, the Frame Ontology defines what can be translated.
If the `Ontolingua` system ensures the translation of domain ontologies described in `Kif` to `Loom`, `Epikit` and `Algernon`, the reverse translation has not been taken into account. Ensuring the translation of `Loom`, `Epikit` and other systems to `Kif`, necessitates the development of new translation tools.

5 Our translation Approach

Our approach to ontology translation [7] aims at using a single translator for all ontology description languages based on description logics, frames or objects. To do this, it makes use of a meta-language named `M-KIF` for the description of representation ontologies and a pivot representation for all the ontologies.

5.1 Representation Ontologies

A description language can be seen as a separate knowledge domain. It has a specific vocabulary in which each term has a unique interpretation. By analogy with an ontology, which is a description of terms that represent a domain or a reality, a representation ontology is the description of the set of representative terms of a language. These are the generic terms of the language which are used to represent the terms of the ontology. Therefore, if a given language `L`$_1$, based on an object formalism, is used to describe an ontology `O`, the representation ontology of `L`$_1$ consists in the specification of the generic terms `Class`, `Attribute`, `Instance`, `ISA` relationship, etc. The description of these terms is independent of their content.

5.2 From `Kif` to `M-Kif`

`Kif` (Knowledge Interchange Format) [3] is an expressive language which plays a similar role to that of data exchange standards, but at a knowledge level. Based on the set theory as defined by Von Neuman-Barnays-Godel[2], it allows formalisation of domain knowledge in terms of objects, relations[3], and functions[4].

`M-Kif` is a language that we have conceived for the description of representation ontologies. As its name indicates, it uses the `Kif` language, with an extension to the `meta-relation` concept.

A meta-relation is a `Kif` relation in which arguments are themselves relations. Only meta-relations with arity ≤ 2 are considered. A `unary meta-relation` is a set of unary or binary relations that satisfy the predicate. A binary meta-relation is a set of couples of unary and/or binary relations that satisfy the predicate. The `M_Kif` definition of a unary meta-relation (respectively a binary meta-relation) captures the semantics of the relations (respectively the couple of relations) that belong to it.

For each representation primitive of a given language we associate a meta-relation with the same name, and its definition using `M_Kif` provides for the capture of the representation primitive semantics .

With the meta-relation concept, we introduce a meta-level description in which the description languages of ontologies can be specified in a declarative and expressive manner.

6 The Pivot Representation

We have identified and defined a core of meta-relations necessary and sufficient for the representation of concepts usually used in styles of DLs, objects, frames

[2] This theory avoids the paradox of set theory.

[3] A `relation` designates a particular relationship between objects of the universe of discourse. It is an arbitrary set of finite lists of objects, where each list is a selection of objects that jointly satisfy the relation

[4] A function is a particular kind of relation in which, each list is such as, the initial elements are arguments, and the final element is the function value when applied on the arguments.

and relations. These are the **predefined meta-relations**. Each one is a predicate of arity ≤ 2, and its extension contains the entities of the source ontology (O_{source}) that satisfy the predicate. We have identified five classes of predefined meta-relations: (1) meta-relations used for the categorisation of O_{source} (**UnaryRelation**, **BinaryRelation** and **Individual**); (2) meta-relations which allow semantics enrichment of O_{source} collections: (**Implicit**, **Explicit**, **Defined** and **Primitive**); (3) meta-relations which allow semantics enrichment of O_{source} associations: (**Domain**, **Range**, **RangeRestiction** , **CardMax**, and **CardMin**); (4) meta-relations which express relationships (other than associations) between O_{source} collections: (**SubRelation** and **Disjoint**); (5) meta-relations which describe links between individuals and collections: (**Has-Instance** and **Is-Instance-Of**).

The pivot representation (PR) of any O_{source} is then represented solely in terms of predefined meta-relation instances.

7 The M_Kif Language

In **M_Kif**, the representation ontology of a given language is a set of definitions that describe the representation primitives (concepts, operators, constructors,...) used by this language. A meta-relation definition associates to the name of the meta-relation, an M_Kif expression that integrates Kif expressions with the following syntax:

```
(defmetarelation name <arguments> := <M_Kif_expression>)
```

For example, the representation ontology of the **Classic** language must contain amongst others, the definition of a meta-relation that specifies the semantics of the **cl-define-concept** operator which introduces a new defined concept. In **M_Kif**, it can be written:

```
(defmetarelation Cl-define-concept (?C1 ?C) :=
      (and (UnaryRelation ?C1)
           (Defined ?C1)
           (Explicit ?C1)
           (ConceptExpr ?C1 ?C)))
```

The ontology representation of **Classic** will be complete when all the meta-relations referenced in every definition are predefined or previously defined.

7.1 Implicit Unary Relations

Each meta-relation conceived for semantic enrichment of binary relations, identifies a particular category of O_{source} collections[5] that we call **implicit unary relations**. These collections do not have a name to designate them explicitly in O_{source}. Consider for example the expression of the meta-relation:

(**Range** R C)

Its extension is a set of couples (R C) such as R $\in$ **BinaryRelation** , C $\in$

[5] They are particular in that they constitute collections which have not been explicitly described as such.

UnaryRelation and the couple (R C) satisfy the **Range** predicate. Each pair (R C) identifies an implicit unary relation, i.e., a set of individuals of O_{source}, that share a single property. This set is formally defined by:

{x / ∀ y: (x y) ∈ R ⇒ y ∈ C}

For example, the corresponding implicit unary relation of the couple (diet vegetable) of the **Range** metarelation determines individuals (not necessarily persons) that share the property "eats vegetables". Each identified implicit unary relation is named by an internal symbol and linked to its category (name of the meta-relation). If IR designates an implicit unary relation, it satisfies the predicates:

(**U**naryRelation IR)
(**I**mplicit IR)
(**D**efined IR)[6]
(**S**ubRelation IR Root)[7]

Each explicit unary relation of O_{source} can then be expressed as a subset of each of the implicit unary relations representing one of its properties. The PR of a O_{source} is therefore brought to a set of explicit and implicit unary relations and a set of binary relations. The unary relations are linked by **SubRelation**.

8 Translation Process

The translation process of O_{source} from a language L_1 to a target one L_2 is entirely driven by O^{rep}_{source} (the representation ontology of L_1) and O^{rep}_{target} (the representation ontology of L_2). The first step consists in building PR, i.e., instanciating the predefined meta-relations. To do that, the translator needs O^{rep}_{source} as an input parameter. which allows it to generate the PR. Fig. 2 illustrates this process.

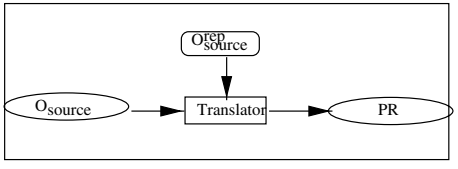


Fig. 2: First step of the translation

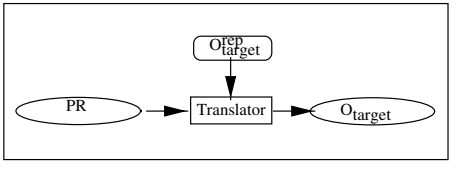


Fig. 3: Second step of translation

Algorithm 1: $O_{source} \Rightarrow$ PR

For each expression (M_i <arguments>) of O_{source}:
1 Search in O^{rep}_{source} for the definition of M_i meta-relation;
2 Apply this definition to (M_i <arguments>)[8]. We obtain:

[6] The description associated to an implicit unary relation represents a necessary and sufficient condition for individuals to belong to this relation.
[7] The pivot representation has a particular unary relation named Root. It is the widest relation in that it includes all unary relations of the source ontology.
[8] This consists in instanciating the second member of the definition with the arguments which appear in (M_i <arguments>).

(M$_{i1}$ arguments)
(M$_{i2}$ arguments)
(M$_{in}$ arguments)

3 If M$_{ij}$ is a predefined meta-relation of the class 3 then
3.1 Generate an implicit unary relation Symb$_k$;
3.2 Describe Symb$_k$;
(**Implicit** Symb$_k$);
(**Defined** Symb$_k$);
(**SubRelation** Symb$_k$ Root);

4 Apply steps **1**, **2** et **3** to all (M$_{ij}$ <arguments>), j=1..n generated in **2**:
5 Stop this process when in all derived expressions (M$_{ij}$ <arguments>), M$_{ij}$ is predefined.

At the end of this phase, the M_Kif translator has built PR. Each explicit unary relation C is expressed as:

(**SubRelation** C X$_i$)

where X$_i$ is an implicit or an explicit unary relation .

The second step of the translation process, illustrated in Fig. 3 is a transformation of PR following the description of the target language included in O^{rep}_{target}. To generate O_{target}, i.e., a specification of O_{source} in a target language, the same translator needs only O^{rep}_{target} as input parameter.

Algorithm 2: PR $\rightarrow$ O_{cible}

For each explicit unary relation C of PR do:
1 For each X$_i$ such as (**SubRelation** C X$_i$)
if X$_i$ is an explicit unary relation
Generate (**Sentence** C X$_i$)
if X$_i$ is an implicit unary relation Symb$_i$
Replace Symb$_i$ by its description;
Generate its associated **Sentence** expression.
2 Apply the second member of the **Sentence** meta-relation definition on all **Sentence** expressions.
Each O^{rep}_{cible} must contain the definition of its specific **Sentence** meta-relation.

9 Example

Consider the following definition expressed in Classic language [1].
`(cl-define-concept 'Vegetarian '(and Person (all diet Vegetable)))`
The defined concept Vegetarian describes persons whose diet is vegetable. Applying Algorithm 1 on Classic O^{rep}_{source} which one definition is given in section 7 enable us to generate PR which contains:

(**SubRelation** Vegetarian Person)

(SubRelation Vegetarian $Symb_0$)

The category of the implicit unary relation $Symb_0$ is **Range** and its value is the couple (diet Vegetable).

If the target language is Osiris [8], its representation ontology must contain definitions of the meta-relations of the Osiris concepts. Among these concepts is the notion of view[9].

In the PR $\rightarrow$ O_{target} direction of the translation, the translator interprets each meta-relation definition as a sufficient condition.

The final result of the translation process is:

```
(View Vegetarian
      ((person)
       (attributes (diet (0,N) Vegetable))
       (assertions ())))
```

10 Conclusion

We have presented an approach to the translation of ontologies using a single translator based on the representation ontologies of the source and target languages. The translator is a program which interprets the meta-relation definitions specified in these representation ontologies.

This translator proceeds in two phases: source to pivot representation PR and PR to target. In the first phase, the translator builds PR of the source ontology, i.e., it instanciates a set of M_Kif predefined meta-relations. In the second phase, the translation consists in transforming PR following the target language's characteristics defined in its representation ontology. The originality of our approach can be situated at three different levels.

Firstly, it needs only one translator whatever the ontology description language. The translator is developed once and for all, and each time a language is targeted for translation, only its representation ontology design using M_Kif is needed.

Secondly, it associates to each language its own ontology, named representation ontology, and therefore each language is considered as a knowledge domain on which principles of ontology design can be applied.

Thirdly, it largely replaces the classical language translation tools by representation ontologies of these languages. We think that writing a representation ontology for a language is easier than writing a specific translator. These two tasks are not situated at the same level: defining a representation ontology is a specification task, whereas writing a translator is a programming task.

The translator is currently under experimentation. The first step was the identification and the implementation of common characteristics of a set of representation languages. Languages based on DLs and Osiris system [8] were retained. Amongst DLs, Classic and Loom were chosen, they are respectively

[9] A view corresponds to a defined or primitive concept in DLs. It is charaterized by the views it inherits, attribute definitions and assertions which represent constraints on attributes.

the least and the most expressive. `Grail` which has been used to write the `CORE` medical ontology, was also targeted. As our work is carried out in the medical domain, it is of particular interest for us. In [6] va have proposed a representation ontology for the `Grail` language.

The aim of this phase is to validate the approach whilst limiting it to two-way translation (source to target and target to source) of a set of common representation primitives to these languages. In a second phase, specific primitives to a given language, which cannot be directly translated, were investigated. Some of these primitives have been identified in [5]. In [7] we have presented a possible treatment of these primitives.

Experimentation of the translator on a real ontology is under way. We have choosen the `Enterprise` ontology [9] which is available on the Ontolingua server as a base for our work. Since this ontology is expressed in the `Ontolingua` language, the design of its representation ontology is also under way. Our aim is to translate the `Enterprise` ontology into the `Classic` and `Osiris` languages.

References

1. Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., & Borgida, A. (1991). Living With Classic: When and How to Use a KL-ONE-like Language, in J. Sowa [ed.], Principles of Semantics Networks: Explorations in the Representation of Knowledge, Morgan Kauffman, San Mateo, CA.
2. Fernández-López, M. (1999). Overwiew of methodologies for building ontologies. Proceedings of the IJICAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden.
3. M. R. Genesereth and R.E. Fikes. (1992). Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Logic Group, Stanford University.
4. T. R. Gruber. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, **5**(2), pp. 199-220.
5. Horrocks, I., R. (1995). A Comparison of Two Terminological Knowledge Representation Systems. Master thesis of Computer Science, University of Manchester.
6. Mihoubi, H., Simonet, A. & Simonet, M. (1998) International Workshop on Description Logics DL'97, Gif sur Yvette, september 27-29, pp132-137
7. Mihoubi, H., Simonet, A. & Simonet, M. (1998) Towards a Declarative Approach for Reusing Domain Ontologies. Information Systems, **23**(6), pp. 355-381.
8. Simonet, A. & Simonet, M. (1995). OSIRIS: an Object-Oriented system Unifying Databases and Knowledge bases. In *Proceedings of KBKS'95: Towards Very Large Knowledge Bases*, Enschede, The Netherlands, N. Mars Ed., IOS Press, pp. 217-227.
9. Uschold, M., King, M., Moralee, S., Zorgios, Y. (1998). The Enterprise Ontology. The Knowledge Engineering Review, Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).
10. Uschold, M., Healy, M., Williamson, K., Clark, P. & Woods, S. (1998). Ontology Reuse and Application. In proceedings International Conference on Formal Ontology and Information Systems, FOIS of 98.

This article was processed using the LaTeX macro package with LLNCS style

A Temporal Object-Oriented Data Warehouse Model

Franck Ravat, Olivier Teste

Université Paul Sabatier - IRIT/SIG
118 Route de Narbonne - 31062 Toulouse cedex 04 (France)
e-mail: {ravat, teste}@irit.fr

Abstract. The data warehousing approach intends to exploit a very large volume of data to make relevant decisions. In this paper, we deal with object-oriented data warehouse design. More precisely, we present an object-oriented data warehouse model, integrating temporal and archive data. We provide functions allowing the administrator to specify a data warehouse from a global source schema.

1 Introduction

Data warehouses (DWs) store large volumes of data, which are extracted from multiple, distributed, autonomous and possibly heterogeneous data sources (operational databases) to improve On-Line Analytical Processing (OLAP) and Decision Support System (DSS) applications [4, 13]. We have specified a functional architecture of a medical DSS[1] in [10]. This architecture (which is divided in three components) distinguishes several issues, laying the foundation for our study.

- The **integration** generates a virtual Global data Source (GS) from multiple data sources. This component is based on federated database approaches.
- The **construction** intends to design and to generate a DW as a materialised view [7] over the GS.
- The **organisation** models data for supporting efficiently OLAP and DSS applications in several Data Marts (DMs). The DMs are often designed according to a multidimensional model [1, 9].

In this paper, we focus on the DW design (construction). We provide means to define a DW from a GS schema. In our DSS, the DW is not organised according to a multidimensional model. We justify our choice by the fact that the multidimensional approach generates a lot of redundant data [4] limiting efficient data management.

1.1 The Problem

The GS is organised to support efficiently management and business activities through OLTP (On-Line Transaction Processing) applications whereas the DW collects relevant information for supporting decision process through OLAP applications. The GS is based on traditional database models to manage business information. On the other hand, the DW must be based on a specific model, related to its requirements, which consist in storing relevant information for decisions. Fig.1 illustrates differences between a GS and a DW.

---

[1] *This study was partially financed by CTI-Sud (computing organism of the French Social Security) located in Midi-Pyrénées (France).It is related to the national REANIMATIC project, which is financed by the French government. This project intends to design and to develop methods and tools for medical DWs.*

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 583–592, 2000.

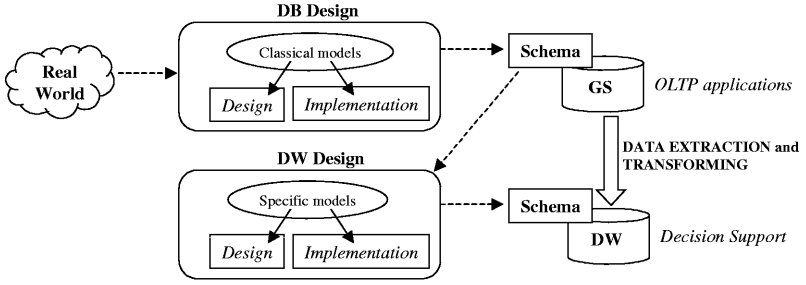


Fig. 1. GS and DW designing.

The main difference lies in the fact that the DW must be modelled from an existing abstract world, which is the GS schema. This requires models for the DW designing, which integrate the following characteristics:

- **Data redundancies** and **data integrity**. In our DSS, the DW is defined as a central repository, which is dedicated to manage relevant decisional information.
- **Complex data**. Our application context is the medical world, where complex information is often used [9].
- **DW data origin**. The DW is populated by source data extractions through extracting and transformation processes.
- **Temporal data**. Many DSS applications are temporal in nature, *e.g.* they are based not only on the up-to-date source information, but also on the history of the data at sources, whereas sources may be non-temporal [4, 14].
- **Historical data with relevant form.** The DW has to limit volumes of historical data ("*recent*" data might be stored with detailed form, whereas "*old*" historical data, which became non-relevant in a detailed form, might be summarised).

One way for dealing with these requirements is to define a specific DW model which integrates the DW characteristics. This paper provides a **temporal object-oriented data warehouse model**. To define the DW, a **building process** is defined.

1.2 Related Work

The DW community focuses on technical aspects such as multidimensional data models [1, 9] as well as materialised view maintenance and configuration [7, 8]. Data warehousing systems have traditionally been used in a relational context such as business area. The relational data warehouse models have limitations due to the nature of the relational data model [3, 9]. Many applications such as medical applications, require more powerful data warehouse models than relational approaches. This can be confirmed by several new proposals in this direction; for example, [3] provides physical implementation using an object-oriented database and [9] defines an extended multidimensional data model for complex data in the clinical world.

Temporal information is very important in a data warehousing context [4, 14]. In [3] the authors do not study temporal aspects of DWs, whereas [9] presents a multidimensional model with temporal characteristics. Our model differs from [9] because we do not organise DW data "*multidimensionally*". Nevertheless, [14] studies incremental maintenance of temporal views using a temporal query language equivalent to TSQL2. We provide more flexible temporal model than [14] because the purging values are not deleted, but they are archived.

Because of its application in data warehousing [13], the materialised view approach has received increasing attention in a relational context [5, 7, 8, 14], whereas little attention is currently paid in the object DW context. Recently [6] provides an object-relational view design for DWs. To build a DW we provide mapping functions for extracting only relevant source data. We assumed that the complete underlying source data are available. In [5] the authors show how to remove warehouse data without fully compute views in a relational framework.

1.3 Paper Outline

Section 2 introduces our temporal model. Section 3 provides a data warehouse model integrating temporal information. Section 4 depicts our prototype.

2 Preliminaries: Handling Time

The temporal dimension is very important in the data warehousing context because DSS applications require historical data [4, 14]. This section introduces the basic temporal concepts for defining several temporal types. Our temporal model is based on a discrete and linear temporal model in which the time line is structured in a "*multigranular*" way [12]. We define a **temporal unit** as a mapping U from the set of positive integers to the set of absolute time sets where, $\forall(i, j)\in\mathbb{N}^2$, $i<j$, $U(i)\neq 0 \wedge U(j)\neq 0 \Rightarrow \forall r_i\in U(i)$, $\forall r_j\in U(j)$, $r_i<r_j$ and $U(i)=0 \Rightarrow U(j)=0$. There is a **finer than** relation among temporal units. Let U_1 and U_2 be temporal units. U_1 is finer than U_2, denoted $U_1\leq U_2$, if for each i, there exists j such that $U_1(i)\subseteq U_2(j)$. **Calendar** is a set of temporal units, which forms a lattice with respect to the *finer than* relation. Moreover, several temporal types are defined.

An **instant** is a time point for a temporal unit $Ins=(n, U)$ where n is a time point and U is a temporal unit. Our model supports several functions between instants, $=$, $\neq$, $<$, $\leq$, $>$, $\geq$ and the following function is defined: $Unit:T\_Instant\rightarrow T\_Unit \mid Unit(Ins)=U$.

An **interval** is the time between two instants, $Int=(InsB, InsE)$ where $InsB\leq InsE$ and $Int=\{Ins \mid InsB\leq Ins\leq InsE\}=[InsB, InsE]$. We adopt the Allen operators (*meets*, *overlaps*,...) and the following functions are defined: $Unit : T\_Interval\rightarrow T\_Unit \mid Unit(Int) = U$, $Begin: T\_Interval\rightarrow T\_Instant \mid Begin(Int) = InsB$, $End : T\_Interval\rightarrow T\_Instant \mid End(Int) = InsE$, $isEmpty : T\_Interval\rightarrow T\_Boolean \mid isEmpty(Int) = true$ *iff* $End(Int) < Begin(Int)$. Note that $isEmpty([Ins,Ins]) = false$.

A **temporal element** h_i is an ordered set of intervals. It is expressed by $h_i = <Int^1; \ldots; Int^h>$ according to the following properties:

- each interval is non empty, $\forall k\in[1..h], isEmpty(Int^k)=false$,
- intervals use a same unit, $\forall k\in[1..h], \forall j\neq k\in[1..h], Unit(Int^k)=Unit(Int^j)$,
- intervals are disjoints, $\forall k\in[1..h], \forall j\neq k\in[1..h], Int^k\cap Int^j=\varnothing$,
- intervals are non contiguous and ordered, $\forall k\in[1..h[, End(Int^k)<Begin(Int^{k+1})$.

3 Data Warehouse Model

A DW stores data, which is extracted from a GS. In our context, GS is described within an object-oriented model. The motivating for using the object-oriented paradigm is that it has proven to be successful in complex data modelling [2].

The case study is taken from the medical domain and it concerns patients and doctors in French hospitals. We use the object definition language provided by Object Database Management Group (ODMG).

```
interface PATIENT(extend Person) {
  attribute Short patient_no;
  attribute String firstname;
  attribute String lastname;
  attribute  Struct  T_Address  {
  String street, String city, Long
  code} address;
  attribute Short birth;
  attribute String sex;
  relationship Set<WARD>
                        admissions;
  relationship Set<CONSULTING>
    consultations
    inverse CONSULTING::patient; }
interface DOCTOR (extend Person){
  attribute Long doctor_no;
  attribute String speciality;
  relationship Set<WARD> works
             inverse WARD::staffs;
  relationship <WARD> manages
         inverse WARD::managed_by;
  relationship Set<CONSULTING>
     consultations
     inverse CONSULTING::doctor; }
interface WARD {
  attribute String title;
  attribute String phone;
  relationship Set<DOCTOR> staffs
             inverse DOCTOR::works;
  relationship <DOCTOR> managed_by
        inverse DOCTOR::manages; }
interface CONSULTING {
  attribute Double fees;
  attribute List<String> symptoms;
  attribute  Struct  T_Pressure  {
  Short     max,     Short     min}
  blood_pressure;
  attribute Double weight;
  attribute Double size;
  attribute Double temperature;
  attribute String diagnosis;
  relationship Set<MEDICINE>
                        prescription;
  relationship <PATIENT> patient
   inverse PATIENT::consultations;
  relationship <DOCTOR> doctor
  inverse DOCTOR::consultations; }
interface MEDICINE {
  attribute String code;
  attribute String mol_category;
  attribute String mol_type;
  attribute Short quantity;
  attribute Double price;
  attribute   String   maker;   }
```

3.1 Warehouse Classes

In order to define the warehouse class concept, we extend the classical class concept of the ODMG with a mapping function for modelling the extracting process as well as two set of properties for specifying the temporal and archive properties through filters.

A **warehouse class** c is defined by ($Name^c$, $Type^c$, $Super^c$, $Extension^c$, β^c, φ^c, ψ^c). $Name^c$ is a unique class name. $Type^c$ is a type defining structure $Structure^c$=($Attributes^c$, $Relations^c$) and behaviour $Methods^c$ (where $Attributes^c$ is a set of attributes, $Relations^c$ is a set of associations and compositions, $Methods^c$ is a set of methods). $Super^c$ is a set of the super classes of c. $Extension^c$=$\{o^1, o^2,\ldots, o^s\}$ is a finite set of warehouse objects. The three specific characteristics are defined as follows:

- β^c is a **mapping function**, which specifies the warehouse class origin,
- φ^c=$\{p_1,\ldots,p_t\}$ is a **temporal filter** defining a set of temporal properties p_i, i∈[1,t] (their detailed value evolutions are kept),
- ψ^c=$\{(agr_1, p_1),\ldots,(agr_a, p_a)\}$ is an **archive filter** defining a set of archive properties p_j, j∈[1,a], which are associated to an aggregate function agr_j.

3.1.1 Warehouse Class Extension ($Extension^c$)

Each extracted source data is stored in the DW as a warehouse object. A **warehouse object** o is defined by (oid, S_0, $History$, $Archive$, $Origin$). oid is an object identity, which is immutable, persists for the lifetime of the warehouse object and it is

independent of the source objects. S_0 is the current state. *History* = $\{S_{p1}, S_{p2},\ldots, S_{pp}\}$ is a finite set of past states. *Archive* = $\{S_{a1}, S_{a2},\ldots, S_{aa}\}$ is a finite set of archive states. *Origin* is a set of source object identities from which the object was created. The warehouse object origin establishes links with source objects. The warehouse object origin allows the DW to periodically propagate source evolutions at the DW level.

A **state** S_i of a warehouse object is defined by a couple (h_i, v_i) where h_i is a temporal element and v_i is a value, which is the object value during instants of h_i. Note that the past states are lost once they are archived. The need for high detailed evolution keeping is in conflict with the need for low volume of data. The administrator manages this problem to keep data evolutions at an acceptable level.

3.1.2 Temporal and Archive Filters (φ^f, ψ^f)

In order to define the past states and the archive states, we introduce the temporal filter concept and the archive filter concept.

A **temporal filter** specifies a set of temporal properties. When a property is temporal, its detailed evolutions are kept. More precisely, at each extracting point (when the DW is refreshed), if the source data has changed, then the old value of the set of the temporal properties is kept into a past state and the current state is refreshed.

An **archive filter** specifies the archive properties (a subset of the temporal properties) and their respective aggregate functions. Several functions are available.

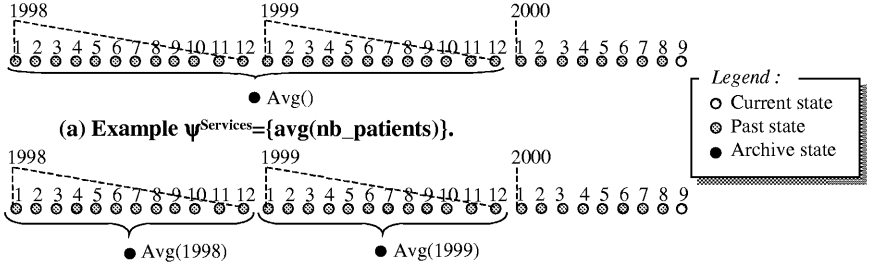
- The classical aggregate functions: *sum*, *count*, *max*, *min*, *avg*.
- Temporal aggregate functions: *sum_t*, *count_t*, *max_t*, *min_t*, *avg_t*. These functions are used to aggregate past values with a temporal criteria.

EXAMPLE: A warehouse class "*Services*" is described as follows:

```
warehouse interface Services { attribute String name; attribute String
specialty; attribute Short nb_patients; }
with temporal filter {nb_patients}, archive filter {avg(nb_patients)};
```

This class is defined from the source class "*WARD*". The temporal filter is composed of one attribute ("*nb_patients*") and the archive filter is composed of one attribute ("*nb_patients*") associated with the average function.

Fig2. illustrates the difference between the *avg* function and the *avg_t* function. We assume that the DW is refreshed every *month*. If the administrator uses the *avg* function (`avg(nb_patients)`) then only one archive state aggregates the temporal states (see Fig 2-a), whereas if he uses the *avg_t* function (`avg(nb_patients,year)`) then one archive state aggregates the temporal states of each *year* (see Fig. 2-b).



(a) Example $\psi^{Services}$={avg(nb_patients)}.

(b) Example $\psi^{Services}$={avg_t(nb_patients, year)}.

Fig. 2. Difference between *avg()* and *avg_t()*.

3.1.3 Mapping Functions (β^c)

Mapping functions define the building process from which the warehouse classes are generated. A mapping function is defined as a composition of basic functions. The basic functions are organised according to several categories, which are depicted in the four following subsections.

(A) STRUCTURING FUNCTIONS: The Structuring Functions (SF) define properties of the warehouse classes from the source properties; SF={π, μ, α}.

- The **project** function $\pi_{\{p1,\ p2,\ldots,\ pm'\}}(cs)=c$ extracts source properties, which are specified with a subset $\{p_1, p_2,\ldots, p_{m'}\}$.
  - $Extension^c=\{o \mid \exists os \in Extension^{cs},\ v_0=[p_1:os.p_1,\ldots, p_m:os.p_m],\ \forall i\in[1,m],\ p_i\in P\}$.
  - $Attributes^c=\{a \mid a\in P \wedge (a\in Attributes^{cs} \vee (a\in Attributes^{cs'} \mid cs'\in Super^{cs}))\}$.
  - $Relations^c=\{r \mid r\in P \wedge (r\in Relations^{cs} \vee (r\in Relations^{cs'} \mid cs'\in Super^{cs}))\}$.
- On the other hand, the **hide** function $\mu_{\{p1,\ p2,\ldots,\ pm''\}}(cs)=c$ specifies a set of non-extracted properties $\{p_1, p_2,\ldots, p_{m''}\}$.
- The **grow function** $\alpha_{\{p1:f1,\ p2:f2,\ldots,\ pm:fm\}}(cs)=c$ creates new properties in a warehouse class. These properties $\{p_1', p_2',\ldots, p_m'\}$ can be calculated attributes or specific attributes. The calculated attributes are defined by an aggregate function $f_i\in\{count, sum, avg, max, min\}$ or a source method $f_i\in Methods^{cs}$. The specific attributes are defined with a simple type $f_i\in\{String, Short, Unsigned\ Short, Float, Double, Boolean\}$ or a complex type $f_i\in\{Set(Type), List(Type)\}$.
  - $Extension^c=\{o \mid \exists os\in Extension^{cs},\ v_0=[p_1:os.p_1,\ldots, p_p:os.p_p,\ p_1':f_1(os),\ldots, p_m':f_m(os)]\}$,
  - $Attributes^c=\{a \mid a\in Attributes^{cs} \vee (a\in Attributes^{cs'} \mid cs'\in Super^{cs})\} \cup \{p_i' \mid p_i'\in P, p_i' \text{ is an attribute}\}$,
  - $Relations^c=\{r \mid r\in Relations^{cs} \vee (r\in Relations^{cs'} \mid cs'\in Super^{cs})\} \cup \{p_i' \mid p_i'\in P, p_i' \text{ is a relation}\}$.

For each warehouse object *o* which is generated through a structuring function, its origin is defined as follows: $Origin(o)=\{os\}$.

EXAMPLE: The administrator creates a warehouse class from the source class "*PATIENT*". He defines the following mapping function.

```
π[p.firstname, p.lastname, p.sex]
        (p α[ nb_consult:count(pp.consultations)](pp PATIENT)
```

The generated warehouse class is composed of the three derived attributes "*firstname*", "*lastname*" and "*sex*". It is also composed of one calculated attributes "*nb_consult*" for explicitly representing the number of consultations of each patient.

(B) POPULATING FUNCTIONS: The populating functions (PF) define source objects that are extracted to populate the warehouse class extensions; PF={σ,⋈}.

- The **select** function $\sigma_p(cs)=c$ restricts the source class extension with a predicate *p*. The generated warehouse class is populated with warehouse objects representing each selected source object according to the selection predicate.
  - $Extension^c=\{o \mid \exists os\in Extension^{cs},\ p(os) \wedge o.v_0 = os.\ v_0\}$, for each warehouse object *o*, its origin is defined as follows: $Origin(o)=\{os\}$.
  - $Attributes^c=\{a \mid a\in Attributes^{cs} \vee (a\in Attributes^{cs'} \mid cs'\in Super^{cs})\}$,
  - $Relations^c=\{r \mid r\in Relations^{cs} \vee (r\in Relations^{cs'} \mid cs'\in Super^{cs})\}$.

- The **join** function $\bowtie_p(cs_1, cs_2)=c$ filters the Cartesian product between two classes according to a join predicate. Each generated warehouse object is defined from a pair of two source objects filtered by the join predicate p.
  - $Extension^c = \{o \mid \exists os_1 \in Extension^{cs1}, \exists os_2 \in Extension^{cs2}, p(os_1, os_2) \wedge o.v_0 = [os_1.v_0, os_2.v_0]\}$, for each warehouse object o, its origin is defined as follows: $Origin(o)=\{os_1, os_2\}$,
  - $Attributes^c = \{a \mid (a \in Attributes^{cs1}) \vee (a \in Attributes^{cs'} \mid cs' \in Super^{cs1}) \vee (a \in Attributes^{cs2}) \vee (a \in Attributes^{cs''} \mid cs'' \in Super^{cs2})\}$,
  - $Relations^c = \{r \mid (r \in Relations^{cs1}) \vee (r \in Relations^{cs'} \mid cs' \in Super^{cs1}) \vee (r \in Relations^{cs2}) \vee (r \in Relations^{cs''} \mid cs'' \in Super^{cs2})\}$.

Note that it is possible to have more than one class as origin class for warehouse classes. If the administrator wants define an attribute from attributes issue from several classes, he must use a join function.

EXAMPLE: We extend the previous example. We select only patients who are adults. Therefore, the generated warehouse class extension is restricted.

```
α[nb_consult:count(pp.consultations)]
 (p π[pp.firstname, pp.lastname, pp.sex, pp.admissions]
                                   (pp σ [q.birth<=1982](q PATIENT)))
```

The association "*admissions*" is projected in the DW. The administrator has to define a warehouse class representing the source class "*WARD*". For example, a warehouse class "*Services*" can be defined through the following function.

```
α[nb_patients:count(vp.patient)] (v π[vp.title]
    (vp ⋈[w∈v.prescription] (v ⋈[vv∈p.consultations](vv CONSULTING, p
σ[pp.birth<=1982](pp PATIENT)), w WARD))
```

(C) SET FUNCTIONS: The set functions (SeF) handle source classes for creating warehouse classes; SeF={$\cup$, $\cap$, $-$, η, η^{-1}}. We provide the classical set functions union, intersect and difference as well as the nest and unest functions. Due to the space limitations we do not detail these functions; a comprehensive definition is given in [11].

EXAMPLE: The administrator extracts information about the prescriptions of the patients using a nest function and a join function. The join function creates a temporary class sharing properties of the source classes "*CONSULTING*" and "*PATIENT*" (restricted to the adults). The nest function regroups some values through the properties "*fees*", "*diagnosis*", "*patient*", "*blood_pressure*", "*weight*" and "*size*".

```
η[g.fees, g.diagnosis, g.patient, g.weight, g.size]::Medicines
    (g π[vp.fees, vp.diagnosis, vp.patient, vp.weight, vp.size,
         vp.code, vp.mol_category, vp.mol_type, vp.price, vp.quantity]
    (vp ⋈[m∈v.prescription] (v ⋈[vv∈p.consultations](vv CONSULTING,p
σ[pp.birth<=1982](pp PATIENT)),m MEDICINE)))
```

(D) HIERARCHISATION FUNCTIONS: The structuring, set and populating functions generate warehouse classes such that $Super^c=\emptyset$. The administrator organises the inheritance relationships between warehouse classes using two hierarchisation functions for generalising and specialising the warehouse classes; HF={Λ, Σ}.

- The **generalisation** function $\Lambda_{\{p1,\ p2,\ldots,\ pm\}}$: $c_1 x c_2 x \ldots x c_n \rightarrow c$ regroups shared common properties and methods. The generated super class c regroups properties of a subset $\{p_1, p_2, \ldots, p_m\}$ (specified by the administrator) of the shared properties between $c_1, c_2, \ldots, c_n$.

- $Extension^c=\{o \mid \exists o' \in \bigcup_{j=1}^{n} Extension^{cj}$, $o.v_0=[p_1:o'.p_1,\ldots, p_m:o'.p_m], p_i \in P\}$, for each generated warehouse object o, $Origin(o)=\{o'\}$.
- $Structure^c=\{p \mid p \in \bigcap_{j=1}^{n} Structure^{cj} \wedge p \in P\}$, $Super^c=\{c \mid c \in \bigcap_{j=1}^{n} Super^{cj}\}$,
- $\forall i \in [1..n]$, $Structure^{ci}=\{p \mid p \in Structure^{ci} \wedge p \notin P\}$, $Super^{ci}=\{c \mid c \in Super^{ci} \wedge c \notin \bigcap_{j \neq i} Super^{cj}\} \cup \{c\}$.

- The **specialisation** function $\Sigma_{\{p1, p2,\ldots, pm\}}$: $c_1 x c_2 x \ldots x c_n \rightarrow c$ creates a subclass from one or several warehouse classes. The result is a subclass c_0 which inherits from the classes $c_1, c_2, \ldots, c_n$; $Super^c=\{c_1, c_2, \ldots, c_n\}$.
  - $Extension^c=\{o \mid \exists o_1 \in Extension^{o1}, \ldots, \exists o_n \in Extension^{on}, p(o_1,\ldots o_n) \wedge o.v_0=[o_1.v_0, \ldots, o_n.v_0]\}$, for each generated warehouse object o, $Origin(o)=\bigcup_{j=1}^{n} Origin(o_j)$.
  - $Structure^c=\{p \mid p \in \bigcup_{j=1}^{n} Structure^{cj}\}$, $Super^c=\{c_1, c_2, \ldots, c_n\}$.

EXAMPLE: The administrator completes the previous example. Fig. 3 depicts the generated data warehouse.

```
Σ[s.specialty="Emergency"](s Services)
```

3.2 Environments

In the previous sections, we have defined the warehouse class concept related to the DW characteristics (detailed or summarised evolutions). This flexible temporal definition of the warehouse classes introduces new problems related to the DW definition. We must deal with heterogeneous data refreshment and heterogeneous warehouse class definition. Our flexible temporal definition of the DW requires a definition of the temporal behaviour (archive threshold,…).

We introduce a new concept, called environment. An environment, which is defined by the administrator regroups warehouse classes having the same temporal behaviour, *e.g.* environments are homogeneous temporal parts (with a specific temporal behaviour). An **environment** env^i is defined by ($Name^{envi}$, C^{envi}, $Config^{envi}$). $Name^{envi}$ is the environment name. $C^{envi}=\{c^{envi}_1,\ldots, c^{envi}_{ni}\}$ is a set of warehouse classes belonging to the environment. $Config^{envi}$ is a set of constraints configuring the environment. The configurations allow the temporal behaviour definitions of each environment. Our approach is based on the ECA-paradigm. We have studied the environment configuration in [10]. Due to the space limitations we do not discuss in detail how the system uses environments; for more details see [10, 11].

EXAMPLE: The administrator creates the environment "*Evolutions*" containing the warehouse classes "*Services*" and "*Patients*"

```
environment Evolutions { Services, Patients }
```

The administrator defines a numeric threshold which limits the past state storage (10 past states). The DW system aggregates the oldest past states in an archive state according to the archive filter definition and the following configuration rule.

```
create constraint NUMERIC_THRESHOLD on environment Evolutions
when self.refresh
if select PS from C in self.classes, O in C.extension, S in O.history
   where S.numerous() > 10
then O.archive(S);
```

4 Implementation

In order to validate our study, we have implemented a prototype called **WarGen**, an acronym of **War**ehouse **Gen**erator, which allows the administrator to define and generate warehouses in an Object-Oriented DataBase Management System (OODBMS). Our prototype is based on the graphical paradigm, *e.g.* we adopt extended UML notations for displaying the GS and the DW schemata. Firstly, the interface displays a graphical representation of the GS schema. Secondly, the administrator defines incrementally a DW schema. Thirdly, the generator creates automatically the DW. The DW schema, the first extraction, which populates the DW and the refreshment process are generated automatically.

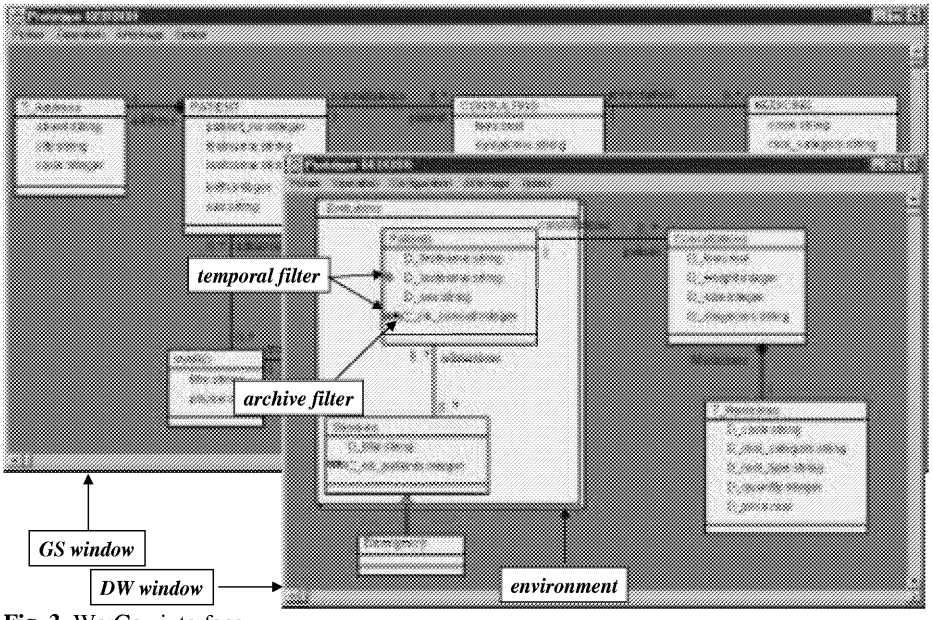


Fig. 3. WarGen interface.

A specific formalism is used to represent environments, temporal filters and an archive filters. Each environment is displayed as a double rectangle containing warehouse classes. The symbol '' defines a property belonging to a temporal filter and the symbol '' defines a property belonging to an archive filter.

5 Conclusion

In this paper, a decision system, which is divided in three components (integration, construction, organisation) is described. This paper deals with the construction component by providing a solution for DW design. More precisely, we provide an object-oriented data model for DWs. We define the warehouse class concept based on the concepts of temporal filter (detailed evolutions) and archive filter (summarised evolutions). In order to define the warehouse class structure, the warehouse class population and the warehouse class hierarchy, we specify mapping functions. In our framework, the data warehouse is generated using functions to specify derived, calculated and specific properties, and organise the inheritance hierarchy of the warehouse classes. Our approach allows administrators to extract only relevant data. An environment defines a temporal part with an homogeneous temporal behaviour (archive threshold,…).

Our solution is implemented in WarGen prototype dedicated to the DW design. The administrator designs graphically, interactively and incrementally a conceptual DW. The generator creates automatically a DW in an OODBMS with the first extracting and the refreshment processes.

There are several possible extensions of this work. In this paper, we depict the class structure extraction through several mapping functions, but the extraction process can be extended to extract the class behaviour. Also, we want to specify a query language for handling our DW elements. This language will be based on temporal extensions of OQL and will integrate graphical elements.

6 References

1. R. Agrawal, A. Gupta, A. Sarawagi, "Modeling Multidimensional Databases", ICDE'97.
2. O.A. Bukhres, A.K. Elmagarmid, "Object-Oriented Multidatabase Systems - A solution for Advanced Applications", Prentice Hall, ISBN 0-13-103813-2, 1993.
3. Buzydlowski J.W., Song I.Y., Hassell L., "A Framework for Object-Oriented On-Line Analytic Processing", DOLAP'98, Bethesda (Maryland, USA), 7 Nov. 1998.
4. S. Chaudhuri, U. Dayal, "An Overview of Data Warehousing and OLAP Technology", ACM SIGMOD Record, 26(1):65-74, 1997.
5. H. Garcia-Molina, W. Juan Labio, "Expiring Data from the Warehouse", VLDB'98.
6. V. Gopalkrishnan, Q. Li, K. Karlapalem, "Star/Snow-Flake Schema Driven Object-Relational Data Warehouse Design and Query Processing Strategies", DAWAK'99, Florence (Italy), August-September 1999.
7. Gupta, I.S. Mumick, "Maintenance of Materialized Views: Problems, Techniques, and Applications", IEEE Data Engineering Bulletin, 1995.
8. Y. Kotidis, N. Roussopoulos, "DynaMat: A Dynamic View Management System for Data Warehouses", ACM/SIGMOD '99.
9. T.B. Pedersen, C.S. Jensen, "Multidimensional Data Modeling for Complex Data", ICDE'99, march 1999.
10. F. Ravat, O. Teste, G. Zurfluh, "Towards the Data Warehouse Design", CIKM'99, 1999.
11. F. Ravat, O. Teste, G. Zurfluh, "A Temporal Object-Oriented Data Warehouse Model - a Comprehensive Definition", Technical Report IRIT/00-14-R, May 2000.
12. X.S. Wang, C. Bettini, A. Brodsky, S. Jajodia, "Logical Design for Temporal Databases with Multiple Granularities", ACM TODS, Vol 22(2), June 1997.
13. J. Widom, "Research problems in data warehousing", ACM CIKM'95, 1995.
14. J. Yang, J. Widom, "Temporal View Self-Maintenance in a Warehousing Environment", EDBT'00, Konstanz (Germany), March 2000.

Process-Oriented Requirement Analysis Supporting the Data Warehouse Design Process
A Use Case Driven Approach

Beate List, Josef Schiefer, A Min Tjoa

Institute of Software Technology (E188)
Vienna University of Technology
Favoritenstr. 9 - 11 / 188, A-1040 Vienna, Austria
{list, js, tjoa}@ifs.tuwien.ac.at

Abstract. A comprehensive requirement analysis for data warehouse systems is mostly often the starting point for the implementation of an enterprise-wide decision support system. Because data warehouse systems concern many organisational units, the collection of unambiguous, complete, verifiable, consistent and usable requirements can be a very difficult task. Use cases are considered as standard notation for object-oriented requirement modelling. We illustrate how use cases enhances communication between domain experts, data warehouse specialists, data warehouse designers and other professionals with different backgrounds. They can be used on a general level, which is intuitive for the users of data warehouse system. This paper explains how use cases can be used to elicit requirements for data warehouse systems, and how to involve the organisational context in the modelling process. With an adapted object model, we demonstrate how to capture different analysis perspectives of the business process. We develop a predefined set of dimension objects that belong to every classic business process and are able to create various fact objects, representing these perspectives.

1 Introduction

Building a data warehouse is a very challenging task because it can involve many organisational units of a company. A data warehouse is a common queryable source of data for analysis purposes, which is primary used as support for decision processes. A data warehouse is multidimensional modelled and is used for the storage of historicized, cleansed, validated, synthesized, operative, internal and external data. Stakeholders of a data warehouse system are interested in analysing their business processes in a comprehensive and flexible way. Mostly they already have a comprehensive understanding of their business processes, which they want to explore and analyse.

What they actually need is a view of their business processes and its data, which allows them an extensive analysis of their data. For this purpose data warehouses are modelled multidimensional, which corresponds to a typical view of its users. This analysis view of the business processes can be very different to the general view even though the underlying process is the same. Hence it is necessary to elicit requirements

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 593-603, 2000.

from the stakeholder of a data warehouse, which belong to their analysis views. The design of data warehouse system is highly dependent on these requirements. Very often data warehouses are built without understanding correctly these needs and requirements and consequently fail for that reason.

The following scenario can often depict the implementation of a new data warehouse system in an enterprise: a system analyst of the IT department or consultant works together with the users to describe the needs and requirements for a data warehouse system. A team of developers receives these descriptions, but they have trouble understanding the business terminology and find the description too informal general to use for implementing the data warehouse system. The developers write their own system specification from a technical point of view. When the system specification is presented to the users, they do not quite understand it because it is too technical. They are, however, forced to accept it in order to move forward.

This approach can easily result in a system that does not meet the requirements of the users because often the users, the system analysts and developers don't speak the same language. Such communication problems can make it difficult to turn a description of an analysis system into a technical specification of a data warehouse system that all parties can understand. In addition, because a technical system specification that is not fully understood by the users, a data warehouse system becomes too difficult or impractical for the intended purposes. Therefore, it will not deliver the expected effect to the company. In these cases often departments will develop data marts for their own purposes, which can be considered as stovepipes and makes an enterprise-wide analysis system impossible.

The challenge is to model a data warehouse system in a way that is both precise and user-friendly. Each symbol describing the analysis process should be intuitive for the user and have defined semantics, so that the developers can use the description as a general, but precise specification of the data warehouse system.

Use cases have two advantages, which make them suitable for representing the requirements for a data warehouse system. First, use case diagrams are part of the UML standard and hence are generally accepted as a notational standard in the software community and second, they can be used on a general level, where implementation details are completely suppressed. Hence, use cases are very suitable for the communication with the users of the data warehouse system.

The IEEE guide [9] states that care should be taken to distinguish between the model for the application and the model for the software, which is required to implement the application model. The model, which is to be used as the basis for the specification is crucial, as it must integrate widely-differing types of information from users. Important characteristics that such a model should possess are [9]:

- **A high level of abstraction.** The model should be at the level of the users' views of the desired system, and should capture their concepts directly. It should not indiscriminately mix this high level information with information that is relevant to lower levels of the development process.
- **Human-readability.** The language in which the model is to be expressed will be used for validating the specification: that is, for presenting the specification to users for their views on its contents. Human understandability is thus the prime concern; and there is a need to avoid the well-known problems that users experience in trying to understand formal requirements specification languages.

- **Precision.** A high-level specification language, for project scope agreement, delineating the system boundaries and naming major objects, rules and processes, is required. Further detail should be left to subsequent development phases. However, the language must be precisely defined, to reduce ambiguity, and to allow for formal integration and consistency checking methods to operate on this representational form.
- **Specification completeness.** It is important that the model captures all aspects of a specification, particularly the information needs of the users.
- **Mappability to later phases.** A requirements definition phase will typically be followed by a detailed analysis and design phases. Therefore, the model should possess a structure suitable for mapping on the later phases.

In the following paper we show an object-oriented approach for a process-oriented requirement analysis, which grasp the stated model characteristics. The remainder of this paper is organized as follows. In section 2 we discuss related works. In section 3 we show how data warehouse requirements arise and their impacts on the data warehouse system. In section 4 we consider the organisational context. Section 5 and 6 show how to apply use cases and object models for the requirement analysis process. The paper concludes with section 7, which presents our current and future work.

2 Related Work and Business Process Orientation

Building a data warehouse is different than developing transaction systems, whereby the requirement analysis process for the latter is supported by numerous methods [2], [4], [6], [12]. Up to now the data warehouse design process has not been supported by a formal requirement analysis method although there are some approaches for requirement gathering. In [19] a catalogue for conducting user interviews in order to collect end user requirements is proposed. Inmon argues in [10] that the data warehouse environment is data driven, in comparison to classical systems, which are requirement driven, and the requirements are understood after it is populated with data and being used by the decision support analyst. He derives the data model by transferring the corporate data model into a data warehouse schema and by adding performance factors. In our opinion, requirements can and must be gathered before the data warehouse design process otherwise only those parts are captured, which are in the basic corporate model.

[1] states that a data warehouse is designed to support the business process rather than specific query requirements, but do not further discuss their statement. An other process driven approach is applied by Kimball [13], whereby the fundamental step of the design process is based on choosing a business process to model. As this approach has proven its success in various projects, and enterprises in general have shifted to process-centred organising, we adopt the process oriented approach for the basis of our work: a formal requirement analysis concept for data warehousing.

The process approach for data warehouse design enables organisations to focus on the performance of business processes and drift away from traditional task or department performance measurement. Hammer also criticised in [8] that everyone is

watching out for task performance, but no one has been watching to see if all the tasks together produce the results they are supposed to, for the customer, the single most important word in the definition business processes.

The change to process centring is not primarily a structural one (although it has deep and lasting structural implications), it is not announced by issuing a new organisational chart and assigning a new set of managerial titles, but process centring is a shift in perspective, in which tasks and processes exchange places [8]. In practice this means that the traditional hierarchical organisation remains and the process view is integrated into the industrial way of organising. Therefore business processes cross organisational boundaries and often tend to be inefficient because of changing responsibilities, long delay times and so forth [15]. Therefore, a process oriented data warehouse design focuses on the entire process including all involved departments and supports a development towards a process-centred organisation.

In [18] we have already started analysing and presenting the requirements for a data warehouse model with a process approach. Use cases and objects were applied as proposed by [12]. We realised that the concept is confined to business processes without any analysis capabilities for decision support purposes and an adaptation to these needs would be required. But we further realised that the support of different views of the models is an opportunity to capture multidimensional and aggregated views of data warehouses.

Basically use case models and object models can be applied to software processes and business processes, which use the same notation but represent different functionality. As argued above, we aim at analysing business processes and focus therefore on the business process approach provided by these models and described in [12]. The use case model describes the business from an external point of view and provides an overview of the area of interest, while the object model is an internal model and describes accurately each business process with its tasks and resources in order to make the model clearer [12]. Use cases represent business processes. In this paper we present the adaptation of the use case and object model in order to support the data warehouse requirement analysis. The aim is to provide a formal requirement engineering method that is easy to use, quickly to understand, covering all major data warehouse model characteristics and is therefore a means of communication between all involved parties in the requirement process. We also apply the model to a classical business process that has already been proven in a real data warehouse environment.

3 Data Warehouse Requirements

Requirements to the data warehouse system determine what data must be available, how it is organized, and how often it is updated. Furthermore the requirements enable the stakeholders to communicate the purpose, establish the direction and set the expectations of information goals for the enterprise. It is advantageous to start the requirement collection process with a macro business discovery (see Figure 1), which is necessary to identify the key business processes, key business metrics, measures and requirements. A high-level model has to be developed depicting how business is conducted. The collection of the requirements of the stakeholder is used to create a vision document, which describes the high-level properties of the data warehouse system. These high-level properties express services, which have to be provided to

meet the requirements of the stakeholders. This relationship between high-level properties of the data warehouse system and its services for the stakeholder leads to a *derivation* of the data warehouse requirements.

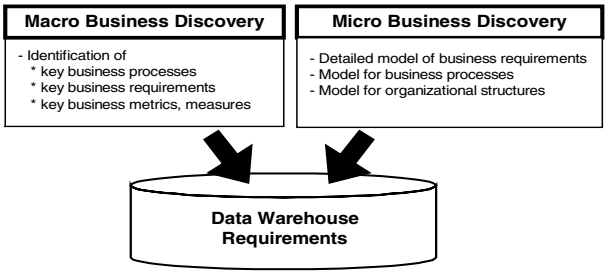


Fig. 1: Requirement Discovery Process for Data Warehouse Requirements

The micro business discovery is an in-depth analysis of the requirements of the organization as defined by the macro business discovery. It describes the business requirements in more detail by considering these requirements in context of the models for the business processes and the organisational structures.

Figure 2 shows the impacts of data warehouse requirements [14]. The figure demonstrates very well, that data warehouse requirements directly affect technical aspects of the data warehouse system. The aim of data warehouse requirements is to provide a comprehensive specification of primary technical aspects of the data warehouse system, to make a successful implementation possible.

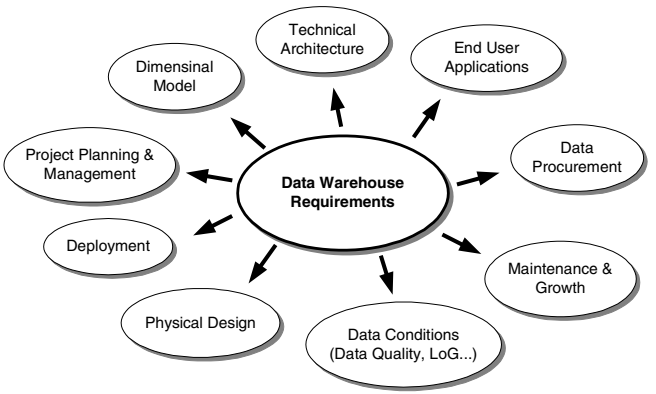


Fig. 2: Impact of Warehouses Requirements (adapted from [14])

In the requirement collection process data warehouse requirements have to be driven by the business requirements. They arise from the macro and micro business discovery and result in a comprehensive technical specification.

Because the business and technical specifications can be very different, business requirements and (technical) data warehouse requirements should be modelled separately. Business requirements describe the needs of the stakeholders from their business point of view. On the other hand data warehouse requirements are derived

from the business requirements and their aim is to provide a comprehensive, precise specification for the data warehouse team. They describe technical data warehouse aspects, whereby the target group is the data warehouse team and not the stakeholders.

Current use case oriented approaches for the requirement analysis are used to describe technical aspects and ignore the business view [5]. Originally use cases were designed to describe primary system behaviour and the involved actors. For data warehousing systems there are other important aspects than system behaviour. Further important aspects are the actual data and information, which are stored in the data warehouse. By integrating this central aspect in the requirement modelling process we present an approach of how to gather the business requirements from stakeholders corresponding to their business processes.

In the requirement analysis process it is important to consider the organisational context of the stakeholders. Furthermore it is important, to classify them in order to provide a comprehensive picture of the future data warehouse users. Next section gives an overview of the stakeholders and their organisational context.

4 Organisational Context

A data warehouse is intended to provide an enterprise-wide decision support for an organisation. It should address the users of all hierarchy levels of the organisation. This broad spectrum of different types of end users requires information at different granularity levels to meet their specific needs.

Figure 3 shows the often-found structure of traditional organizations. Within such a structure, different requirements for data processing and data analysis can be identified, which correspond to the different layers of the organisation. As we move upward through the layers of the hierarchy, for example, the level of granularity required decreases. Executives use highly summarized information, while line managers work at a detailed level. Administrative users need to create and maintain individual data items such as orders or customer records. Professional users and line managers require statistical analysis tools. Executives require systems that highlight anomalies or geographically show key indicators, and allow drill-down in problem areas.

During the micro business discovery requirements of data warehouse users of all different levels in the hierarchy are identified. This discovery process should allow a combination of all requirements to model a coherent enterprise-wide decision support system. What data warehouse users require is an environment that allows data coherence and functional integration, enabling them to achieve their business goals.

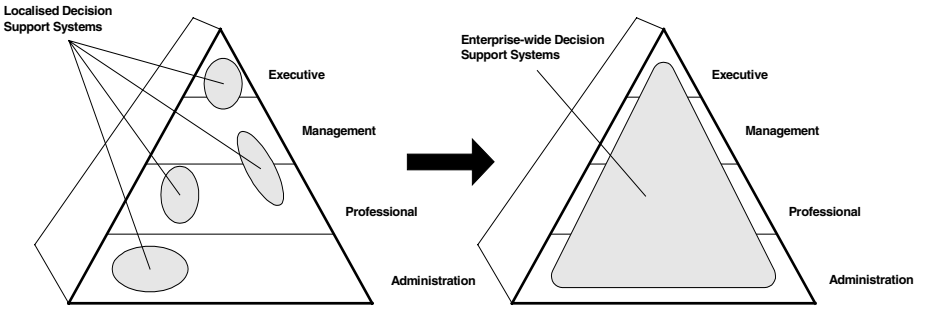


Fig. 3: Localised vs. Enterprise-wide Decision Support Systems

Figure 3 shows an enterprise-wide decision support system with its organisational hierarchy, which can achieve these business goals. For gathering the meshed requirements of such a decision support system, it is necessary to use models which represent all business requirements on each hierarchy level and which allow a requirement consolidation of all levels.

5 Use Case Model

The use case model captures business processes in the company that satisfy the customers' interests, and the interests of others outside the company (partners, suppliers, etc) [12]. Our use case model describes business processes in the company that are analysed and provide information to the data warehouse user and the company's staff. The model shows how analysts use the data warehouse. It provides therefore a specification of the data warehouse user's roles and the business processes they are analysing.

Analysis System and Subsystems. The analysis system or subsystem is the modelling concept we use to symbolise the business or area of responsibility that we analyse. We use the same symbol as Jacobson in 0, the rectangle with rounded corners.

Analyst. The analyst represents a role that someone or something in the environment can play in relation to the analysis system. In our model analyst represent the environment that analyse business processes belonging to the specified business system.

Use Case. The use case is our construct for a business process that is analysed.

Uses Association. The uses association is a must association and describes use cases that are aggregated into other use cases. This concept must be applied when parts of the use case are analysed by other analysts.

5.1 Example

We have applied our entire data warehouse use case model in the example of **Fig. 4**. The analysis system is a grocery store with a warehouse and a market as subsystems. The use case 'Managing Demand' represents the business process that receives the goods in the warehouse, manages the storage in the warehouse, delivers the goods to the supermarket and manages the storage in the market until they are sold. This use case is analysed by the fulfilment manager and aggregates the managing order use case and the selling use case. We have applied a uses association to managing order and selling, because managing demand combines and therefore aggregates both use cases. The managing order use case belongs to the warehouse and is analysed by the logistic manager and the selling use case belongs to the market and is analysed by the supermarket head.

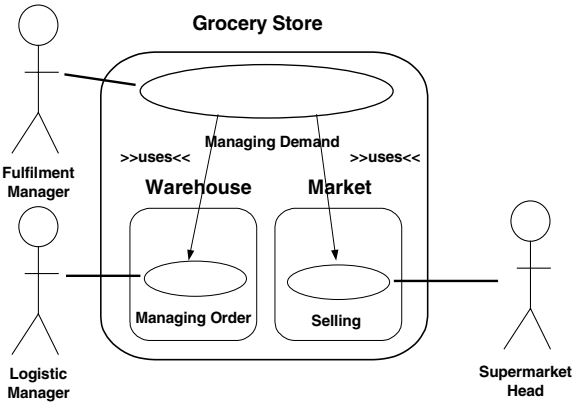


Fig. 4: Example of an Analysis System

6 Object Model

The object model provides a clear picture how the use case is structured internally in order to realise the analysis capabilities of the process (backward engineering) and to describe the analysis requirements of various actors (forward engineering). The internal structure consists of fact and dimension objects.

6.1 Dimension Objects

Business processes have a lot of characteristics in common. Analyses of various traditional and fully accepted business process definitions provide various process attributes that reflect the structure of classical business processes. As we apply an object-oriented paradigm to business processes in general, and the object model to the internal structure, we have to change our point of view and look for objects in the business process instead of attributes. Describing processes by the means of objects

gives more importance to the internal structure and additionally empowers to become an equal part of the process.

Davenport states in [3] that a process is a structured, measured set of activities designed to produce a specified output for a particular customer or market. He notes that a business process is a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure of action. Hammer argues in [7] that a business process is a group of tasks that together create a result of value to the customer. A business process is a collection of activities that takes one ore more kinds of input and creates an output that is of value to the customer. Jacobson's approach in [12] is, that the purpose of each business process is to offer each customer the right product or service (that is, the right deliverable), with a high degree of performance measured against cost, longevity, service and quality. Davenport's process definition comprises measure, activity, customer, market, time, place, input and output objects. Hammer's definition consists of activity, result, customer and input objects, and Jacobson focuses on the customer, product, service, deliverable and measure objects. Davenport provides the most comprehensive definition, which basically includes the objects of both other definitions.

The objects in the process definitions highlight key business process characteristics, but represent also classical data warehouse dimensions (e.g. organisation, customer, product, service, time, etc.) and data warehouse facts (measure). As these key business process objects can be found in any classical business process, we propose a standard set of dimension objects representing data warehouse dimensions (Figure 5):

Fig. 5: Data Warehouse Main Dimension Objects in Classic Business Processes

Customer Dimension Object. The single most important word in the definition of process is 'customer' and a process perspective on a business is the customer's perspective [8]. Our strong focus on business processes, with the satisfied customer as the main target of the process, leads directly to the customer dimension object.

Deliverable Dimension Object. Davenport and Hammer do not specify the output [3] or result of value [7] of the process in their definition, while Jacobson's approach is to offer the right product or service to the customer [12] and integrates both terms into deliverable. The customer does not see or care about the company's organisational structure or its management philosophies; the customer sees only the company's products and services, all of which are produced by its processes [8]. As the output or result of the value of a process might be the opposite of the defined process deliverables, and products or services are too specific for a requirement analysis model, we cover the process targets with the deliverable dimension object.

Organisation Dimension Object. Business processes typically flow through several organisational units and cross a lot of responsibilities. Leymann and Roller argue in [15] that business processes that often cross organisational boundaries tend to be inefficient because of changing responsibilities, long delay times, etc. and further that

it is obvious that the process reflects the hierarchical structures of the organisation. The analysis of the organisational structure detects occurrences that are caused by certain units and therefore organisation dimension object is required.

Time Dimension Object. A business process is a specific ordering of work activities across time with a beginning and an end [3]. The end of the process represents a point in time where the results of the process are delivered and cannot be changed anymore. We address this point for the time dimension object.

Fact Object. Performance is measured against cost, longevity, service and quality [12]. These measures (e.g. turnover, profit, ratio rates etc.) represent facts in data warehouse notation and fact object in this paper. As business processes often consist various facts, which are created by different dimensions, we introduce a communication association to demonstrate the communication - the fact - that is created by certain dimension objects (see Figure 6).

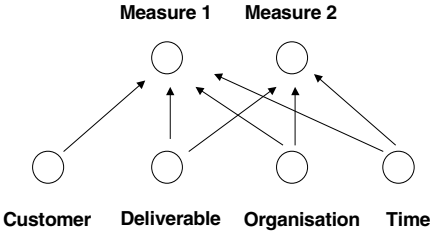


Fig. 6: Data Warehouse Fact Objects

7 Conclusion

In this paper we presented the adaptation of use case and object models for modelling business requirements for data warehouse systems to support the data warehouse design process. We showed how data warehouse requirements are derived from business requirements and their organisation context.

Our use case model is an excellent means of both expressing requirements with regard to the data warehouse and providing a comprehensive picture of what the data warehouse is intended to perform. It illustrates the function of the business, the process analysts and the business process to be analysed with its aggregations.

The object model is an internal model that captures different analysis perspectives of the business process. Various facts represent these perspectives, which are influenced by dimension objects.

In our future work we will concentrate improving the discussed models in order to support more appropriately the data warehouse design process.

References

[1] Anahory, S. Murray, D., *Data Warehousing in the real World – A practical Guide for Building Decision Support Systems*, Addison-Wesley Publishing 1997.

[2] Coad, P., Yourdon, E., *Object-Oriented Analysis. Englewood Cliffs*, N.J.: Prentice Hall 1991.

[3] Davenport, Th., *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press 1993.

[4] Davis, A., Hisa, P., *Status Report: Requirements Engineering*, IEEE Software November 1993.

[5] Debevoise, N. T., *The Data Warehouse Method*, Prentice-Hall, 1999.

[6] Finkelstein, A., *Requirements Engineering Research: Coordination and Infrastructure*, Requirements Engineering 1, 1996.

[7] Hammer, M., Champy, J., *Reengineering the Corporation – A Manifesto for Business Revolution*, Harper Collins Publishers 1993.

[8] Hammer, M., *Beyond Reengineering*, Harper Collins Publishers 1996.

[9] IEEE, *IEEE Guide to Software Requirements Specifications*, IEEE Inc., 1984

[10] Inmon, W. H., *Building the Data Warehouse*, Wiley & Sons 1996.

[11] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., *Object-Oriented – Software Engineering – A Use Case Driven Approach*, Addison Wesley 1992.

[12] Jacobson, I., Ericson, M., Jacobson, A., *The Object Advantage – Business Process Reengineering with Object Technology*, ACM Press, Addison-Wesley Publishing 1995.

[13] Kimball, R., *The Data Warehouse Toolkit: Practical Techniques For Building Dimensional Data Warehouse*, John Wiley & Sons 1996.

[14] Kimball, R., Reeves, L., Ross, M., Thornthwaite, W., *The Data Warehouse Lifecycle Toolkit*, John Wiley & Sons, 1998

[15] Leymann, F., Roller, D., *Production Workflow – Concepts and Techniques*, Prentice Hall PTR, 2000.

[16] List, B., Schiefer, J., Tjoa, A. M., Quirchmayr, G., *The Process Warehouse - A Data Warehouse Approach for Business Process Management*, In e-Business and Intelligent Management - Proceedings of the International Conference on Management of Information and Communication Technology (MiCT1999), Copenhagen, Denmark, 1999.

[17] List, B., Schiefer, J., Tjoa, A. M., Quirchmayr, G., *The Process Warehouse Approach for Inter-Organisational e-Business Process Improvement*, To appear in Proceedings of Re-Technologies for Information Systems (ReTIS2000) integrated in Reengineering Week 2000, Zurich, Switzerland, 2000.

[18] List, B., Schiefer, J., Tjoa, A. M., Quirchmayr, G., *A Generic Data Model for the Process Warehouse - An Approach for Multidimensional Business Process Analysis*, Proceedings of Business Information Systems 2000 (BIS 2000), Poznan, Poland, 2000.

[19] Poe, V., *Building a Data Warehouse for Decision Support*, Prentice Hall 1995.

On Data Warehouse and GIS Integration

Zdeněk Kouba, Kamil Matoušek, Petr Mikšovský

The Gerstner Laboratory for Intelligent Decision Making and Control
Faculty of Electrical Engineering
Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
Phone: (+420 2) 2435 7379, Fax: (+420 2) 2435 7224,
{kouba, matousek, miksovsp}@labe.felk.cvut.cz

Abstract. A system composed of a data warehouse (DWH) integrated with geographical information system (GIS) yields significant advantages and introduces a new decision making support quality. The advantages, drawbacks and implementation strategies of the integrated system are discussed.

This paper attempts to identify necessary requirements for functionality of the integration module in order to provide complete and consistent system interconnection. The solution should be general enough to be easily re-implemented for a large family of GIS and DWH. Within the project, special attention should be dedicated to the methodology of spatial data aggregation. It is necessary to keep in mind that two different types of information can be used to describe spatial data, namely geometric properties (location, area etc.) and topological ones.

A software component, which interconnects these two technologies, is under development within the GOAL research project[1].

1. Introduction

Currently, there are several promising and efficient information technologies for decision support, such as data warehouses (DWH) and geographical information systems (GIS). A research challenge is the integration of these sophisticated technologies. Such an integrated system will introduce a new quality into decision making support. Managerial personnel will be supported in decision-making as usually by their aggregated enterprise information. Moreover, they acquire access to relevant geographical information stored in the appropriate linked geographical data store. Typical geographically-oriented analytical queries will be processed by GIS analytical kernel. E.g. searching for all the customers living by a specified river, road

[1] This research is partially supported by project INCO-COPERNICUS No. 977091 GOAL (Geographical Information On-Line Analysis) and grant of Czech Ministry of Education No. OK380

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 604–613, 2000.

etc. From the GIS prospective, geographical data is enriched by time dimension and fast access to the huge amount of historical data. The performance of repeated demanding GIS queries is improved by means of a data warehouse. *Integration module* – the software component which interconnects the two technologies is under development within the GOAL research project.

It is necessary to keep in mind that two different types of information can be used to describe spatial data, namely geometric properties (location, area etc.), and topological ones (position of two objects can be described e.g. as adjacent or as inclusion if one object is inside the other one, etc.). With respect to this fact, there should be developed specialized means for:

- data aggregation utilizing characteristic features of the domain (e.g. geographic object hierarchy),
- data presentation stressing the spatial relations, and
- knowledge discovery in databases with geographical context.

To achieve a tight coupling of a GIS and a Data Warehouse, the architecture of such an integrated system is roughly introduced.

- The particular instance of the *GIS* part can be represented by a commercial GIS system. Necessary modification for establishing appropriate communication of other modules with the GIS will be suggested. Data security aspects belong to the topics that will be studied and implemented in the future. The other important sub-modules incorporate the data mining methods for knowledge discovery in databases. They help to answer the decision-maker's "what happens if ..." questions. In data mining, it is desirable to modify current machine learning methods for spatial data mining.
- *Integration module*: The aim of related efforts is not just integration of two particular software products. A detailed study of general problems related to the integration of such software products is the main objective of the research. An interesting theoretical problem is the incremental update of materialized spatial views.

2. Integration Module

There are two main roles of the integration module. One of them is the transformation of the data from external data sources. The other one concerns with the integration of GIS and DWH components of the overall system.

The aim of the integration module is to co-ordinate actions carried out by the GIS system and the data warehouse. A motivating software tool showing some aspects of such behavior has been developed. It enables to evaluate possible alternatives of GIS – data warehouse integration and co-ordination.

General integration architecture of the two systems – DWH and GIS – contains seven basic building blocks enabling the functionality:

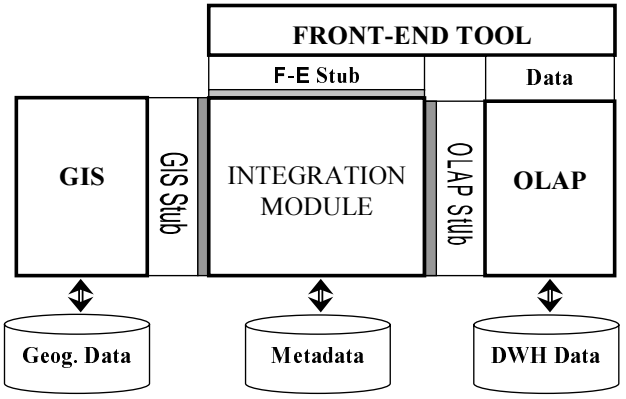


Fig.1. GIS-DWH Integration Architecture

- **Core Integration Module -** application independent, metadata [2] driven integration tool interconnecting DWH and GIS using an optional dimensional navigator. Universal general interfaces for DWH, GIS and dimensional navigator Stubs are being designed.
- **OLAP Subsystem -** OLAP (data warehouse) system. It is connected to the integration module by the application-specific stub.
- **GIS Subsystem -** geographical information system. It is connected to the integration module by the application-specific stub.
- **Front-End Tool -** generic navigation tool represents a minimal set of necessary functions including dimensional navigation and data filtering. Front-end tool is an optional component the existence of which is necessary only in the case that its functionality is not performed within GIS or DWH subsystems. Example of such a front-end software can be Microsoft Excel 2000.
- **OLAP Stub -** implementation dependent adjustment of the universal OLAP interface to the application-specific requirements.
- **GIS Stub -** similarly, implementation dependent adaptation of the universal GIS interface to the application-specific requirements. The GIS specific scripting tools can be utilized.
- **Front-End Stub -** implementation dependent adjustment of the universal front-end tool interface to the application-specific requirements. Front-end stub is an optional component.

2.1 OLAP Component

Data warehouses are aimed at providing very fast responses to user queries. Usually, these queries are generated by OLAP layer of the data warehouse architecture [1]. The data model of a data warehouse is designed to support fast evaluation of very

complicated OLAP queries. Let us define the following toy example for explaining the basic concepts of data warehouse modelling.

Consider an imaginary grocery chain consisting of nine supermarkets located in three districts in Bohemia. The districts are *Eastern Bohemia*, *Central Bohemia,* and *Western Bohemia*. Let the data warehouse store the data on turnover of particular supermarkets.

The data warehouse structure is represented by a cube, dimensions (axes) of which are *time*, *assortment*, and *location*. Every elementary cell of the cube contains a real number identifying the turnover achieved by the corresponding supermarket (position along the *place* axis) in respective time slot (position along the *time* axis), and particular assortment item (position along the *assortment* axis). This number is called *fact*, *measure,* or *metric*. Let us choose the term *fact* to avoid ambiguity.

2.2 GIS Component

For the system integration purposes we consider an object - oriented [3] GIS model distinguishing these basic GIS elements:

- *GIS objects* - represent individual data items (points, lines, areas etc. with attached sets of data values).
- *GIS classes* - abstract groups of objects of the same type (level), e.g. region, district, building, etc.

It is not necessary to map each internal GIS class to geographical metadata class and vice-versa. In general, the GIS system should be able to manipulate with the whole geographical data under examination. For example, the GIS system need not know the details about the *geographical class hierarchy*. Its structure is stored in a metadata object, accessible directly from the integration module.

In most cases, the GIS system is considered to be a geographical data source providing the rest of the system with geographical data. A subset of this data is then retrieved and offered to the data warehouse.

The object model can have various *data representations* based on the particular GIS capabilities:

- A straightforward representation could be based on *layers* (in some GIS systems called *themes*) for *GIS classes* and individual *layer objects* (sometimes called *features*) as *GIS objects*.
- In the case of a particular object-oriented GIS system, its *GIS classes* and *objects* can be defined independently on the structure of GIS layers.

Concerning the example defined in previous paragraph, GIS representation corresponding to the problem consists of common geographical elements. There is an area labeled as *region* (*Bohemia*). Inside the *region* there are *districts* and *cities*.

It is possible to implement the concept using various programming skills and technologies. Inter-process communication technologies of a particular operation

system can be utilized for communication between GIS systems and Integration module.

For ensuring generality of the relationship between GIS and OLAP, the integration methods are evaluated using various GIS systems. The first one, *GT Media 98,* is an object-oriented information system, which represents a solution for many kinds of potential users with import and export capabilities to other GIS formats. On the other hand the second system, ArcView by ESRI, is a relatively cheap and commonly well-known system worldwide. Each of these two systems offers completely different communication interfaces, therefore the integration module has to support various types of communication. This key difference is overcome using the concept of well-defined stubs within the integrated system.

2.3 Integration Aspects

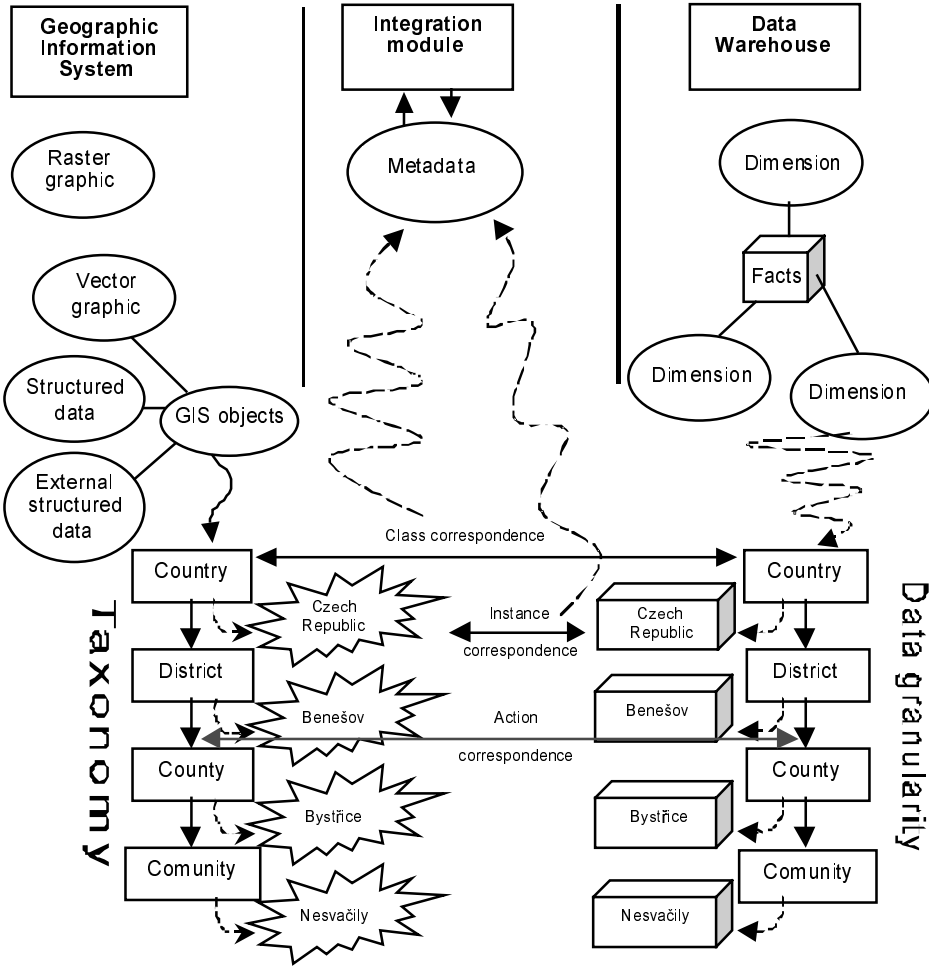
A data warehouse model is represented mainly by the structure of its dimensions, dimensional hierarchies, facts and pre-defined aggregations. For each aggregation level there exist several instances.

Geographical information system stores data in various formats including raster graphics (e.g. map backgrounds) and vector graphics (various object shapes, networks etc.). There exist several kinds of *GIS objects*. They are related to the vector graphical representation, structured data (physical tables, logical – in-memory relational tables) and external structured data sources. In particular, for a specific GIS application there exists taxonomy of GIS objects, which can be used for geographical navigation.

The binding elements between GIS and DWH are the elements of the data warehouse *location* dimension and the GIS objects' taxonomy. The task of the integration is to provide three kinds of necessary dynamic correspondences:

- *Class correspondence* - maps particular aggregation levels of the *location* dimension to the corresponding GIS taxonomical levels and vice versa. This is relatively long-time static pre-defined information stored in integration metadata.
- *Instance correspondence* - maps particular instances of aggregation levels to the instances of the "classes" in sense of the previous item and vice versa. This is a more dynamic part of the integration information and it guaranties the run-time data integrity. Integration module should keep track of the instance changes in both GIS and DWH and propagate them to the other sub-system.
- *Action correspondence* - this is the most dynamic correspondence, which ensures navigation consistency. E.g. after changing aggregation level using particular front-end tool, status information has to be changed in integration module and propagated to the data warehouse and GIS. Another example: when performing a GIS selection, just particular sub-cube should be considered for calculation, i.e. corresponding filter should be applied in the data warehouse and the front end tool should show it.

The concept is depicted in Fig 2. Taxonomical structure of GIS objects is bound with aggregation levels of a data warehouse via class correspondence. Correct mapping of particular objects is guarded by class correspondence (GIS object "Czech Republic" is related to the instance "Czech Republic" of an aggregation level *country* of the *location* dimension in the data warehouse). The arrow labeled action correspondence represents a response in case the location aggregation level changes



from "District " to "County". Metadata objects support all of the three correspondences.

Fig.2. Schema of a DWH – GIS integration module

3. Integration Module Architecture

As for the GIS and Front-End tools, there are a great variety of systems being more or less adaptable while on the other hand we have many black box – like commercial solutions. The concept of stubs enables to develop the inter-process layer as "thin" as possible for achieving fast performance and as "thick" as necessary to implement all the required interface functionality.

Status variables contain all the information necessary to identify current status of data browsing. Basically it contains selection and filter information. With the use of this set of variables, the synchronization of particular modules is centralized.

3.1 GIS Stub

Basic functionality of the GIS stub component can be described by the following set of functions. Some of them are required for the GIS stub functionality, while some are not mandatory (the ones marked by an empty circle).

Refresh GIS: Updates GIS using the status information from Integration Module state variables.

Refresh Integration Module: Updates Integration Module status variables using the GIS information.

Show Results: This executive function uses the state information in the integration module state variables, invokes the desired query in the data warehouse and displays the resulting table grid in the GIS environment.

Return Instances for Instance Level: Retrieves all the corresponding instances for a given aggregation level but only those, which are co-incident with a given instance of a certain level.

Init: GIS stub initialization. Called by GIS or in some cases by Integration Module when initializing interconnection through this GIS stub.

Done: Releasing GIS stub from memory. Called by GIS or in some cases by Integration Module when closing up interconnection through this GIS stub.

Most of these functions are to be called by GIS (except the "*Return...*" functions). In case insufficient GIS extensibility, all the communication can be Integration Module driven.

3.2 Front-End Stub

Basic functionality of the front-end stub component can be described by a following set of functions. Some of them are required for the front-end stub functionality, while some are not mandatory (the ones marked by an empty circle).

Refresh FE: Updates front-end using the status information from Integration Module state variables.

Refresh Integration Module: Updates Integration Module status variables using the front-end information.

Show Results: This executive function uses the state information in the integration module state variables, invokes the desired query in the data warehouse and displays the resulting table grid in the front-end environment.

3.3 OLAP Stub

In most cases, the data warehouse front-end tools have native ability to communicate with OLAP and data warehouse part of the integrated system. This is achieved using specialized data / OLAP drivers (also called providers) and often hidden technologies with specific (if any) interfaces. When integrating such a system, the OLAP stub part can be realized as "OLAP Hooker" – a tool, which percepts the communication flow between the front-end tool and OLAP server and informs the integration module about the changes in data scope for purposes of synchronization of the system. In this case, all the requests from the integration module to the data warehouse are first sent to the front-end tool, which propagates them to the OLAP.

3.4 Typical Operations

There are two basic operation modes of the entire system. One of them concerns the typical data browsing and navigation using visual tools, while the other lays in data gathering and filling into the system.

- Data navigation using GIS and / or front-end tool. Typically, the "*Refresh ...*" and "*Show Results*" stubs' functions are invoked.
- Data transformation. In this case, GIS component serves as a geographical data source and via GIS stub it yields the required new information. The "*Return ...*" functions of the GIS stub are typically used for dependency recognition.

3.5 Dimensional Navigator Concept

As one of the possible front-end tools the simplified "dimensional navigator" can be taken into account. It is a specialized software component, which assists the user in the data view selection. It is an approach that enables using of some proprietary GIS systems with weaker scripting support or extends maximal achievable functionality within the proprietary GIS system.

In both cases, the two operations should be supported and reflected in the underlying integrated sub-systems:

- Aggregation level selection (roll-up and drill-down operations)
- Applying a filter (constraint selection)

While in the data warehouse this means to set the current granularity level and to choose the optimal aggregated data chunks to be used, in the GIS the particular selection is marked by a different element color, shape texture or outline. In each

moment, the current class, instance and level *configuration* is stored in integration module status variables, which has to be continuously and efficiently updated.

4. Motivating Case Study

The work on the integration module specifications introduced the need for a verification of selected concepts under investigation. To study related methods and algorithms a simple motivating prototype was developed.

The effort was also motivated by authors' willing not to be dependent on one specific GIS system (GT Media 98), but to support more general framework. Particularly, the ArcView GIS system has been used for the first demonstration of cooperation between GIS and data warehouse (Microsoft SQL Server 7 with OLAP Services) on the data of a case study application – Water distribution tracking [4]. ArcView has been chosen also thanks to its simple scripting ability.

There are several choices in the ArcView system for interoperability with external programs, namely Remote Procedure Call (RPC) and Dynamic Data Exchange (DDE) or using shared memory accessed via DLL (Dynamic Link Library). Moreover, the scripting is supported using internal ArcView language *Avenue*. For the purposes of our application we needed both to develop the graphical user interface (dimensional navigators) and to support database access. The use of shared memory for the communication was chosen as an optimal solution. In our motivating example, the Dimensional Navigator represents a tool enabling user interaction with the data warehouse dimensional structure (Dimensional Navigator Panes, Measure Pane) and the tool for retrieving calculated tabular results from the underlying data in the data warehouse.

The *Avenue* scripts were written to achieve the following tasks: **selection by** circle **shape**, rectangular shape and general polygonal shape. The required feature to select pipes connected to specific reservoir (in the water distribution application) has been implemented utilizing previously mentioned functions in combination with **selection mode switch** (search for the nearest or included element / search for all pipes in the surroundings of the selected reservoirs).

Each of the selection scripts generates a list of the selected data items and properties. Passing this list, the communication with external integration tool is accomplished.

Interactive graphical navigator was developed to enable the user control of the cooperation between GIS objects and data warehouse contents.

5. Future Development

Currently, the efficient implementation constrained by given fixed dimensional hierarchy has been developed. Next steps will be oriented at overcoming this

restriction. Utilization of spatial indexing techniques [5], [6] will enable the system to translate spatial queries into OLAP queries even for arbitrarily specified areas.

6. Summary

The paper presents first ideas concerning integration of geographical information systems with data warehouses. The authors designed first version of a general framework for integration of such systems. The described integration is based on two layers: data, where *class* and *object* correspondence was defined, and *actions*, where both systems have to keep track on actions being performed. During the research, two different GIS systems were taken to account to ensure independence on a specific software product. The verification of these ideas was done using a motivating case study, although there are still many open research problems in this area including e.g. the data security concepts of the integration module.

Currently, the implementation of the integration module is targeted to the Microsoft Windows platform. A special version of OLE-DB [7] provider for OLAP (Microsoft SQL Server OLAP Services), which allows "listening" to MDX queries (Multidimensional Extensions to SQL by Microsoft) sent to the server was developed. These queries are then parsed and the result is used for system synchronization, while reducing the necessary scripting efforts.

References

1. Kurz A.: *Data Warehousing – Enabling Technology*, (in German), MITP-Verlag GmbH, Bonn, 1999, ISBN 3-8266-4045-4
2. Rauber A., Tomsich P.: TR2 – Metadata language specification, Technical report INCO-COPERNICUS 977091 “GOAL”, 1999
3. Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorenson W.: *Object-oriented Modelling and Design*, Prentice-Hall, Englewood Cliffs., New Jersey, 1991
4. Matoušek K., Svoboda L.: *Extending GIS by Data Warehouse*, In: Proc.of Int.Carpathian Control Conference 2000, TU Košice, 2000, pp. 339-342, ISBN 80-7099-510-6
5. Guttman A.: R-trees: a dynamic index structure for spatial searching. Proc. of SIGMOD Int. Conf. on Management of Data, 1984, pp. 47-54
6. Beckmann N. et al.: *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*. Proc. of 1990 ACM/SIGMOD Int. Conf. on Management of Data, May, Atlantic City, USA, 1990, pp. 23-25
7. *Microsoft OLE-DB 2.0 Programmers Reference and Data Access SDK*, Microsoft Press, 1998, ISBN 0-7356-0590-4

Semi-automatic extraction of hyponymies and overlappings from heterogeneous database schemes

Luigi Palopoli, Domenico Saccà[†], Giorgio Terracina, Domenico Ursino[†]

Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria
Via P. Bucci, 87036 Rende (CS), Italy
{palopoli,sacca,terracina,ursino}@si.deis.unical.it
[†] This author is also supported by ISI-CNR

Abstract. This paper tackles the problem of semi-automatically extracting hyponymy and overlapping properties between entities belonging to pre-existing database schemes. The algorithm we propose consists of two phases: the first one derives basic hyponymies and overlappings starting from a particular scenario; the second one takes in input basic properties derived by the first phase and extracts further, more general hyponymies. In addition the paper shows some applications of derived hyponymies and overlappings.

1 Introduction

In order to foster the re-engineering and innovative utilization of large information sources available today, ad-hoc approaches have been devised, including Cooperative Information Systems [8, 16, 19] and Data Warehouses [4, 18].

It is known that integration algorithms, needed for synthesizing Cooperative Information Systems and Data Warehouses, can benefit from the availability of derived *interscheme properties* [1–3, 5, 7, 9, 12, 14]. Among them, the most significant ones are the following:

- *Synonymies*: a synonymy indicates that two objects have the same type[1] and the same meaning;
- *Homonymies*: an homonymy tells that two objects have the same name and the same type, but different meanings;
- *Type Conflicts*: a type conflict holds between two objects if they denote the same concept, but have different types;
- *Hyponymies/Hypernymies*: given two entities E_1 and E_2, E_1 is a *hyponym* of E_2 (which is, in its turn, the hypernym of E_1) if E_1 has a more specific meaning than E_2. As an example, the entity *PhD_Student* is a hyponym of the entity *Student*. If both E_1 and E_2 belong to the same scheme (resp., to

[1] The type of an object indicates if it is an entity, a relationship or an attribute; object types are sometime called "meta-types" in the literature.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 614–623, 2000.

different schemes) we say that E_1 is an intrascheme (resp., an interscheme) hyponym of E_2 and that E_2 is the intrascheme (resp., the interscheme) hypernym of E_1;
- *Overlappings*: given two entities E_1 and E_2 an overlapping property exists between them if there exist non-empty sets of attributes $\{A_{11}, A_{12}, \ldots, A_{1n}\}$ of E_1 and $\{A_{21}, A_{22}, \ldots, A_{2n}\}$ of E_2 such that, for $1 \leq i \leq n$, A_{1i} is a synonym of A_{2i};
- *Object Cluster Similarities*: an object cluster is a set of connected objects in a scheme, i.e. a subscheme; this kind of properties thus indicates similarities between connected subschemes;

Recently, several proposals appeared in the literature dealing with deriving synonymies and homonymies [1–3, 5, 7, 9, 14]; some papers tackle the problem of detecting type conflicts and object cluster similarities [7, 12, 13, 15, 17] and some attempts have been carried out for obtaining uniform techniques deriving almost all kinds of properties (e.g., techniques of [7] derive synonymies, type conflicts and object cluster similarities, whereas those of [12, 17] detect synonymies, homonymies, type conflicts and object cluster similarities). The problem of deriving hyponymies and overlappings received less attention; moreover, to the best of our knowledge, the papers describing methods for deriving hyponymies, such as [3, 7, 10], require a rather significant intervention of the human expert; finally, to the best of our knowledge, no techniques for deriving overlappings have been proposed in the literature.

Deriving hyponymies and overlappings is important in order for the (relative) semantics of input database schemes to be correctly reconstructed. For instance, [3, 14] show how reliable scheme object similarities can be obtained, once a set of hyponymy properties is available. In addition [1, 3, 7, 11] show how scheme integration can benefit of the availability of synonymies, distinctnesses and hyponymies.

In this paper we present a two-phase technique for semi-automatically deriving hyponymies and overlappings.

The first phase derives the so-called basic hyponymies and overlappings; we call them basic because they are valid only in the following scenario. Consider a scheme S_1 and suppose that, here, E_i is an intrascheme hyponym of E'_i. In addition, consider a scheme S_2 and suppose that it contains E_j that is an intrascheme hyponym of E'_j. Finally, assume that E'_i and E'_j are synonyms. If all these conditions hold, $[E_i, E_j]$ is called a *candidate pair*. The first phase aims at determining the kind of relationship existing between objects E_i and E_j. Indeed, there are four kinds of relationships that may hold between elements of candidate pairs $[E_i, E_j]$, namely *(i)* E_i and E_j are synonyms, *(ii)* E_i and E_j are distinct, *(iii)* E_i is a hyponym of E_j or vice versa, *(iv)* E_i and E_j partially overlap, i.e., there are non-empty sets of attributes $\{A_{i1}, A_{i2}, \ldots, A_{in}\}$ of E_i and $\{A_{j1}, A_{j2}, \ldots, A_{jn}\}$ of E_j such that, for $1 \leq k \leq n$, A_{ik} is a synonym of A_{jk} (we denote the set of these attributes as *overlapping attributes*).

The second phase of the algorithm takes in input basic hyponymies, derived by the first phase, and synonymies, derived by applying some of the algorithms

for finding synonymies presented in the literature, such as those illustrated in [1–3, 5, 7, 9, 12, 14], and obtains further, more general, hyponymies.

The second phase is based on the consideration that a new hyponymy can be derived from pre-existing synonymies and hyponymies only if one of the following transitivity rules can be applied:

1. A hyponymy property holds from an entity E_i to an entity E_j and a further hyponymy exists from E_j to E_k, in which case a hyponymy can be derived to hold from E_i to E_k;
2. A hyponymy property holds from E_j to E_k and a synonymy holds between E_i and E_j, in which case a hyponymy can be derived to hold from E_i to E_k;
3. A hyponymy property holds from E_i to E_j and a synonymy exists between E_j and E_k, in which case a hyponymy can be derived to hold from E_i to E_k.

The algorithm we present here exploits some thresholds and weighting factors. Their values were set and tuned empirically, by conducting a series of experiments. In particular we have used the database schemes of the Italian Central Governmental Offices (ICGO) as our main benchmark for tuning and validation. Similarly to what we have done for fixing analogous weighting factors we have adopted within other interscheme property extraction algorithms [13, 14, 12], in order to derive correct values of coefficients for weights and thresholds, we have selected a small subset of ICGO databases and we have run our algorithm on it several times, using different values for coefficients. Final coefficient values have been then validated over several further database schemes, and proved to be well tuned except for some minor adjustments.

The plan of the paper is as follows. The next section describes in all details the proposed algorithm; Section 3 describes some applications of hyponymies and overlappings; finally, in Section 4, we draw our conclusions.

2 Extracting hyponymies and overlappings

As previously pointed out, our algorithm consists of two phases: the first one derives basic hyponymies and overlappings whereas the second one derives more general hyponymies. In the next sections we describe these two phases.

2.1 Phase 1

Phase 1 of the proposed algorithm takes a set of schemes S and a Starting Synonymy Dictionary SSD as the input and yields in output the four dictionaries $IHSD$, $IHDD$, $IHHD$ and $IHOD$. In the following we will describe into details how each of these dictionaries is constructed. All the functions we introduce in the following will refer to a pair of schemes $S_1, S_2 \in S$. In order to obtain all properties they must be applied to any pair of schemes $S_p \in S, S_q \in S, p \neq q$.

Constructing the Synonymy Dictionary First consider the following preliminary definitions:

- For any value $th \in [0,1]$, let $SSD_{th} = \{\langle O_i, O_j, f\rangle \in SSD \mid f > th\}$.
- $CandS$ denotes the set of candidate pairs of S_1 and S_2, i.e., the set of pairs $[E_i, E_j]$ such that *(i)* E_i is an intrascheme hyponym of E'_i in S_1, *(ii)* E_j is an intrascheme hyponym of E'_j in S_2 and *(iii)* the triplet $\langle E'_i, E'_j, f\rangle$ belongs to SSD_{th_s}. Here th_s is a suitable threshold; its value has been experimentally set to 0.52 (see the Introduction for details on experiments).

Then the Dictionary of Synonymies between Intrascheme Hyponyms can be derived as follows:

$$IHSD = \{\langle E_i, E_j, f_{E_iE_j}\rangle \mid ([E_i, E_j] \in CandS) \wedge (\langle E_i, E_j, f_{E_iE_j}\rangle \in SSD_{th_s})\}$$

In plain words, $IHSD$ contains pairs of intrascheme hyponyms stored, in their turn, as synonyms in the SSD and having a "high" plausibility coefficient.

Constructing the Hyponymy Dictionary In order to define the $IHHD$, we need the following preliminary definitions:

- $CandH$ denotes the set of candidate hyponym pairs and can be obtained as $CandH = \{[E_i, E_j] \mid ([E_i, E_j] \in CandS) \wedge (\langle E_i, E_j, f\rangle \notin IHSD)\}$;
- Let OS_1 and OS_2 be two sets of objects and TS be a generic set of triplets $\langle O_i, O_j, f\rangle$ such that $O_i \in OS_1$ and $O_j \in OS_2$. Then $\mu_{\langle TS,OS_1,OS_2\rangle}$ denotes a matrix with a row (resp., column) for each object O_i in OS_1 (resp., O_j in OS_2) and $\mu_{\langle TS,OS_1,OS_2\rangle}[O_i, O_j] = f$ if $\langle O_i, O_j, f\rangle \in TS$, $\mu_{\langle TS,OS_1,OS_2\rangle}[O_i, O_j] = 0$ otherwise.
- The function $\delta(F, P, Q, \omega_v)$ returns a factor obtained from computing the objective function of a maximum weight matching, as explained next. The input here are: *(i)* two sets of objects $P = \{p_1, \ldots, p_n\}$ and $Q = \{q_1, \ldots, q_m\}$, *(ii)* a weight matrix F on P and Q such that, for each $p_i \in P$ and $q_j \in Q$, $0.0 \leq f_{ij} \leq 1.0$, and *(iii)* a coefficient ω_v. The output is a value in the real interval $[0,1]$. If $P = \emptyset$ or $Q = \emptyset$, then $\delta(F, P, Q, \omega_v)$ returns 0. Otherwise, let $BG = (P \cup Q, A)$ be a bipartite weighted graph, where A is the set of weighted edges $\{(p_i, q_j, f_{ij}) \mid f_{ij} > 0\}$; the maximum weight matching for BG is a set $A' \subseteq A$ of edges such that for each node $x \in P \cup Q$ there is at most one edge of A' incident onto x and $\phi(A') = \left(\sum_{(p_i,q_j,f_{ij}) \in A'} f_{ij}\right)$ is maximum (for algorithms solving maximum weight matching, see [6]). Now, let $\overline{\phi}(A') = \frac{\phi(A')}{|A'|}$. The value returned by $\delta(F, P, Q, \omega_v)$ is:
$$\delta(F, P, Q, \omega_v) = \left(1 - \tfrac{1}{2}\omega_v \times \frac{abs(|P|-|Q|)+2\times(min(|P|,|Q|)-|A'|)}{|P|+|Q|}\right) \times \overline{\phi}(A')$$
In the formula above, the ratio denotes the fraction of objects unrelated in the constructed matching w.r.t. their total number. We have experimentally set ω_v to 2 (see the Introduction for details about the experiments).
- The function $\delta'(F, P, Q)$ is analogous to the function $\delta(F, P, Q, \omega_v)$ but it computes the objective function of the corresponding maximum weight matching without taking into account the presence of arcs which do not participate into the matching. In particular, if we consider the definitions of BG, A' and $\phi(A')$ introduced for δ, the function δ' is defined as $\delta'(F, P, Q) = \frac{\phi(A')}{|A'|}$;

- $Att(E)$ denotes the set of attributes of the entity E, whereas $AttH(E)$ denotes $Att(E)$ plus the set of attributes of E's intrascheme hypernym.
- The set $\sigma_{\langle E_i,E_j\rangle}$ includes all triplets $\langle A_1, A_2, f\rangle$ in SSD_{th_α} such that $A_1 \in Att(E_i)$ and $A_2 \in Att(E_j)$; as usual, th_α denotes a threshold value in the real interval $[0,1]$; we have experimentally set th_α to 0.55.
- The value $\delta_\sigma(E_i, E_j)$ is computed by determining the objective function associated to the maximum weight matching of a bipartite graph whose nodes are the attributes of E_i and E_j and whose edges have a weight equal to the synonymy coefficient existing between the corresponding attributes. In the computation of the objective function, the number of attributes which do not participate into the matching is taken into account:
$$\delta_\sigma(E_i, E_j) = \delta(\mu_{\langle\sigma_{\langle E_i,E_j\rangle},Att(E_i),Att(E_j)\rangle}, Att(E_i), Att(E_j), \omega_v)$$
- The value $\delta'_\sigma(E_i, E_j)$ is analogous to $\delta_\sigma(E_i, E_j)$ except that, in the computation of the result, it considers only the attributes participating to the matching. More formally we have:
$$\delta'_\sigma(E_i, E_j) = \delta'(\mu_{\langle\sigma_{\langle E_i,E_j\rangle},Att(E_i),Att(E_j)\rangle}, Att(E_i), Att(E_j))$$
- $PotH$, the set of potential interscheme hyponyms of S_1 and S_2, is defined as the set of pairs $[E_i, E_j]$ being candidate hyponym pairs and having a value of $\delta_\sigma(E_i, E_j)$ greater than a certain threshold. More formally $PotH$ can be expressed as $PotH = \{[E_i, E_j] \in CandH \mid \delta_\sigma(E_i, E_j) > th_l\}$, where th_l is a threshold value in the real interval $[0,1]$; th_l has been experimentally set to 0.25.
- Let AS and AS' be two sets of attributes and TS a set of triplets of the form $\langle A, A', f\rangle$, where A and A' are attributes and f is a plausibility coefficient. Then $AS \subseteq_{TS} AS'$ if $(\forall A \in AS)(\exists A' \in AS')(\langle A, A', f\rangle \in TS)$. $AS \subset_{TS} AS'$ if $(AS \subseteq_{TS} AS')$ *and* $(AS' \not\subseteq_{TS} AS)$.

We are now in the condition of defining $IHHD$, as follows:

$$\begin{aligned} IHHD = \{\lceil E_i, E_j, \delta'_\sigma(E_i, E_j)\rceil \mid \\ ((([E_i, E_j] \in PotH) \wedge (AttH(E_j) \subset_{SSD_{th_\alpha}} AttH(E_i)) \vee \\ (([E_j, E_i] \in PotH) \wedge (AttH(E_j) \subset_{SSD_{th_\alpha}} AttH(E_i)))\} \end{aligned}$$

The rationale underlying this definition is the following. Given an intrascheme hyponym E, all attributes of E's intrascheme hypernym are also attributes of E. E_i is an interscheme hyponym of E_j if all attributes of E_j and of E_j's intrascheme hypernym are similar to attributes of E_i or of E_i's intrascheme hypernym but at least one attribute of E_i or of E_i's intrascheme hypernym is not similar to any attribute of E_j or of E_j's intrascheme hypernym.

Constructing the Overlapping Dictionary Also in this case, we need to introduce some preliminary concepts:

- NHD denotes the set of candidate pairs $[E_i, E_j]$ such that E_i and E_j are neither synonyms nor hyponyms. It is defined as follows:
$$NHD = \{[E_i, E_j] \mid ([E_i, E_j] \in PotH) \wedge (\lceil E_i, E_j, f\rceil \notin IHHD)\}$$

- The set $\nu_{\langle E_i,E_j\rangle}$ includes triplets of the form $\langle A_1, A_2, f'\rangle$, where $A_1 \in AttH(E_i)$, $A_2 \in AttH(E_j)$ and either $(A_1 \in Att(E_i))$ or $(A_2 \in Att(E_j))$, and f' is a coefficient derived from the (sufficiently high) synonymy coefficient associated to $[A_1, A_2]$ in the SSD_{th_α}. Formally:
$$\nu_{\langle E_i,E_j\rangle} = \left\{ \begin{array}{l} \langle A_1, A_2, f'\rangle \mid A_1 \in AttH(E_i), A_2 \in AttH(E_j), \\ \qquad (A_1 \in Att(E_i) \vee A_2 \in Att(E_j)), \langle A_1, A_2, f\rangle \in SSD_{th_\alpha}, \\ \qquad f' = \left\{ \begin{array}{ll} f & if\ A_1 \in Att(E_i) \wedge A_2 \in Att(E_j) \\ c_\nu \times f & otherwise \end{array} \right. \end{array} \right\}$$
Since synonymies involving an attribute belonging to an intrascheme hypernym provide a minor contribution to the final decision than those where both involved attributes belong to intrascheme hyponyms, the factor c_ν $(0 \leq c_\nu \leq 1)$ is used to normalize the corresponding coefficient; the value of c_ν has been experimentally set to 0.7 (see the Introduction for details about these experiments).
- The value $\delta_\nu(E_i, E_j)$ is computed by determining the objective function associated to the maximum weight matching of a bipartite graph constructed from the attributes of E_i and E_i's hypernym, on the one hand, and of E_j and E_j's hypernym on the other hand. More formally we have:
$$\delta_\nu(E_i, E_j) = \delta(\mu_{\langle \nu_{\langle E_i,E_j\rangle}, AttH(E_i), AttH(E_j)\rangle}, AttH(E_i), AttH(E_j), \omega_v)$$
Therefore each pair of matching nodes represents a pair of synonym attributes; the set of attributes associated to pairs of matching nodes constitutes the set of *overlapping attributes*.

We are finally in the condition to define $IHOD$, as follows:
$$IHOD = \{\|E_i, E_j, \delta_\nu(E_i, E_j)\| \mid ([E_i, E_j] \in NHD) \wedge (\delta_\nu(E_i, E_j) > th_d)\}$$
where th_d is a threshold value in the real interval $[0, 1]$; we have experimentally set th_d to 0.33.

Constructing the Distinctness Dictionary The dictionary $IHDD$ is defined as:
$$\begin{array}{l} IHDD = \{\langle\langle E_i, E_j, 1 - f_{E_iE_j}\rangle\rangle \mid (\langle E_i, E_j, f_{E_iE_j}\rangle \in SSD) \wedge \\ \qquad (([E_i, E_j] \in CandH) \wedge ([E_i, E_j] \notin PotH)) \vee \\ \qquad (([E_i, E_j] \in NHD) \wedge (\|E_i, E_j, g\| \notin IHOD)))\} \end{array}$$
Intuitively, $IHDD$ contains all candidate pairs $[E_i, E_j]$ such that E_i is to be considered distinct from E_j. Since there are two kinds of entity diversities, the set $IHDD$ consists of two subsets:
- entity pairs $[E_i, E_j]$ such that they are yielded as candidate entity pairs by $CandH$ but they are completely distinct, i.e., their attributes are so different that the function δ_σ returns a value under a certain (low) threshold.
- entity pairs $[E_i, E_j]$ such that they are not completely different (as those contained in the first subset) but their overall similarity, computed taking into consideration also the attributes of their intrascheme hypernyms, is low enough that neither a synonymy, nor a hyponymy nor an overlapping exists between them.

It is worth pointing out that the coefficient associated to distinctness depends on the synonymy coefficient. Note that, if we use an approach based both on syntax (i.e., object attributes) and on semantics (i.e., object neighborhoods) for deriving the synonymy coefficient (such as one among those of [1–3, 5, 7, 9, 12, 14]), also the distinctness coefficient will depend both on syntax and on semantics.

2.2 Phase 2

The Hyponymy Dictionary, yielded in output by Phase 2, is constructed by a function γ; formally: $IHHD = \gamma(\langle S, SSD, IHHD\rangle)$.

The function γ consists mainly in the computation of the fixpoint of a function Ψ, and is encoded as follows:

$$\gamma(\langle S, SSD, IHHD\rangle) = \pi_2\left(\Psi^{\infty}\left(\langle SSD, \zeta(S, IHHD)\rangle\right)\right)$$

where π_2 denotes the projection on the second component of a tuple.

The function $\zeta(S, IHHD)$ takes in input a set of schemes S and a Hyponymy Dictionary $IHHD$, storing basic hyponymies existing between objects belonging to schemes of S. The function enriches the $IHHD$ by adding a tuple for each intrascheme hyponymy holding in S; each added tuple has a plausibility coefficient equal to 1.

The fixpoint associated to the function Ψ is defined as follows:

$$\begin{cases} \Psi^0(\langle SSD, HD\rangle) = \langle SSD, HD\rangle \\ \Psi^i(\langle SSD, HD\rangle) = \Psi(\Psi^{i-1}(\langle SSD, HD\rangle)) \quad for\ i > 0 \end{cases}$$

The functor Ψ takes in input a Synonymy Dictionary and a Hyponymy Dictionary and enriches this latter one by deriving new properties from the set of pre-existing ones. It yields in output the Synonymy Dictionary, as it was received in input, and the Hyponymy Dictionary, enriched with the new derived properties.

Since three cases exist leading to the derivation of new hyponymies (see above, the Introduction) the functor Ψ can be encoded as follows:

$$\Psi(\langle SSD, HD\rangle) = \langle SSD, HD \cup \xi_1(\langle SSD, HD\rangle) \cup \xi_2(\langle SSD, HD\rangle) \cup \xi_3(\langle SSD, HD\rangle)\rangle$$

The function ξ_1 (resp., ξ_2 and ξ_3) manages the first (resp., the second and the third) of these cases. It can be encoded as follows:

$$\xi_1(\langle SSD, HD\rangle) = \varphi(\theta_1(\langle SSD, HD\rangle))$$

$$\theta_1(\langle SSD, HD\rangle) = \{\lceil E_i, E_k, \tau(f_{E_iE_j}, f_{E_jE_k})\rceil \mid (\lceil E_i, E_k, f\rceil \notin HD) \wedge (\lceil E_i, E_j, f_{E_iE_j}\rceil \in HD) \wedge (\lceil E_j, E_k, f_{E_jE_k}\rceil \in HD)\}$$

In the function above, for computing the coefficient associated to the new derived hyponymy, we use a fuzzy operator $\tau(\alpha, \beta)$. The operator represented by $\tau(\alpha, \beta)$ is a t-norm [5]. We recall that t-norms are dyadic functions from $[0, 1] \times [0, 1]$ to $[0, 1]$ that are monotonic, commutative and associative and have been used to define fuzzy set intersection. We shall restrict ourselves to the following t-norms:

$\tau_1(\alpha, \beta) = max(0, \alpha + \beta - 1)$	*exclusive*
$\tau_2(\alpha, \beta) = \alpha \times \beta$	*independent*
$\tau_3(\alpha, \beta) = min(\alpha, \beta)$	*inclusive*

It can easily be shown that $\tau_1(\alpha, \beta) \leq \tau_2(\alpha, \beta) \leq \tau_3(\alpha, \beta)$ holds for $0 \leq \alpha, \beta \leq 1$, and so τ_1 is the most "pessimistic" t-norm whereas τ_3 is the most "optimistic" one. From the experimental results (see the Introduction) we have determined that the most appropriate t-norm to be applied in the function θ_1 (as well as in functions θ_2 and θ_3 below) is $\tau_2(\alpha, \beta)$.

The function φ takes in input a set of triplets denoting hyponymies and discards the weakest ones, i.e., those having a plausibility coefficient under a certain threshold. It is as follows:

$$\varphi(T) = \{\lceil E_i, E_j, f_{E_iE_j}\rceil \mid (\lceil E_i, E_j, f_{E_iE_j}\rceil \in T) \wedge (f_{E_iE_j} > th_\varphi)\}$$

th_φ has been experimetally set to 0.6.

Functions ξ_2 and ξ_3 manage the second and the third case for deriving new hyponymies we have described in the Introduction; they can be encoded as follows:

$$\begin{aligned}
\xi_2(\langle SSD, HD\rangle) &= \varphi(\theta_2(\langle SSD, HD\rangle)) \\
\theta_2(\langle SSD, HD\rangle) &= \{\lceil E_i, E_k, \tau(f_{E_iE_j}, f_{E_jE_k})\rceil \mid (\lceil E_i, E_k, f\rceil \notin HD) \wedge \\
&\quad (\langle E_i, E_j, f_{E_iE_j}\rangle \in SSD) \wedge\ (\lceil E_j, E_k, f_{E_jE_k}\rceil \in HD)\} \\
\xi_3(\langle SSD, HD\rangle) &= \varphi(\theta_3(\langle SSD, HD\rangle)) \\
\theta_3(\langle SSD, HD\rangle) &= \{\lceil E_i, E_k, \tau(f_{E_iE_j}, f_{E_jE_k})\rceil \mid (\lceil E_i, E_k, f\rceil \notin HD) \wedge \\
&\quad (\lceil E_i, E_j, f_{E_iE_j}\rceil \in HD) \wedge\ (\langle E_j, E_k, f_{E_jE_k}\rangle \in SSD)\}
\end{aligned}$$

It should be clear, from the definition of θ_1, θ_2 and θ_3, that the derivation of a hyponymy at the step i of the fixpoint could lead to the possibility of deriving new hyponymies at the step $i+1$. Therefore the fixpoint terminates when, during one iteration, no new hyponymies are found.

The number of steps of the fixpoint is polynomial in the number of scheme objects since the computation of the fixpoint is equivalent to the transitive closure of a suitable graph.

3 Applications

As previously pointed out, hyponymies and overlappings can be applied in many contexts such as *(i)* the extraction of more reliable scheme object similarities [3, 14]; *(ii)* the correct reconstruction of the semantics of input database schemes [2, 3]; *(iii)* a higher quality scheme integration and abstraction [1, 3, 7, 11]. In the following we illustrate the third of these applications.

Generally speaking there are four kinds of interscheme properties that must be considered by an algorithm for scheme integration; each kind of interscheme properties leads to a different representation of the involved objects in the global scheme, which is constructed by the integration algorithm. In more detail, given two schemes S_1 and S_2 to integrate, the possible cases are:

- *Two entities $E_i \in S_1$ and $E_j \in S_2$ are synonyms*, in which case the global scheme S_{12} contains a unique entity representing both E_i and E_j.
- *E_i and E_j are distinct*, in which case both E_i and E_j are present in S_{12}.
- *E_i is a hyponym of E_j*, in which case the global scheme contains both E_i and E_j but E_i will be an intrascheme hyponym of E_j.
- *E_i and E_j partially overlap*; remember that, in this case, there are non-empty sets of attributes $\{A_{i1}, A_{i2}, \ldots, A_{in}\}$ of E_i and $\{A_{j1}, A_{j2}, \ldots, A_{jn}\}$ of E_j such that, for $1 \leq k \leq n$, A_{ik} is a synonym of A_{jk}. Moreover remember that an overlapping property between E_i and E_j can hold only if *(i)* E_i is an intrascheme hyponym of an entity E'_i, *(ii)* E_j is an intrascheme hyponym of an entity E'_j and *(iii)* E'_i and E'_j are synonyms (and, therefore, have been integrated into an entity E'_{ij}). In this case the global scheme contains two entities E''_i and E''_j obtained from E_i and E_j by eliminating the overlapping

attributes; the two entities E_i'' and E_j'' are completely distinct. If E_i'' and E_j'' are the only hyponyms of E_{ij}', an attribute A_k $(1 \leq k \leq n)$ is added to E_{ij}' for representing overlapping attributes A_{ik} and A_{jk}. Vice versa, if E_i'' and E_j'' are not the only hyponyms of E_{ij}' a new entity E_{ij}'' is created; E_{ij}'' is an intrascheme hyponym of E_{ij}' whereas E_i'' and E_j'' are intrascheme hyponyms of E_{ij}''; for each k, such that $1 \leq k \leq n$, an attribute A_k is added to E_{ij}'' for representing overlapping attributes A_{ik} and A_{jk}.

4 Conclusions

In this paper we have presented an algorithm for the extraction of hyponymies and overlappings from database schemes. The algorithm consists of two phases: Phase 1 derives basic hyponymies and overlappings, Phase 2 extracts more general hyponymies.

As the current work, we are adding the prototype module implementing the techniques presented here in a more general and complex system called DIKE (Database Intensional Knowledge Extractor) aimed at supporting the activities of both a designer of a Cooperative Information System and a Data Warehouse. In particular, the system *(i)* extracts interscheme properties from the set of databases into consideration; *(ii)* defines scheme semantics by exploiting extracted interscheme properties; *(iii)* integrates and abstracts schemes taking into account their semantics; *(iv)* constructs the Cooperative Information System or the Data Warehouse.

ACKNOWLEDGMENTS.
The authors gratefully acknowledge the Italian Information System Authority for Public Administration (AIPA) for kindly providing ICGO schemes and technical support with them.

References

1. C. Batini and M. Lenzerini. A methodology for data schema integration in the entity relationship model. *IEEE Transactions on Software Engineering*, 10(6):650–664, 1984.
2. M.W. Bright, A.R. Hurson, and S. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database Systems*, 19(2):212–253, 1994.
3. S. Castano and V. De Antonellis. Semantic dictionary design for database interoperability. In *Proc. of International Conference on Data Engineering (ICDE'97)*, pages 43–54, Birmingham, United Kingdom, 1997. IEEE Computer Society.
4. S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *ACM SIGMOD RECORD*, 26(1):65–74, 1997.
5. P. Fankhauser, M. Kracker, and E.J. Neuhold. Semantic vs. structural resemblance of classes. *ACM SIGMOD RECORD*, 20(4):59–63, 1991.
6. Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18:23–38, 1986.

7. W. Gotthard, P.C. Lockermann, and A. Neufeld. System-guided view integration for object-oriented databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(1):1–22, 1992.
8. A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of 22th International Conference on Very Large Data Bases (VLDB'96)*, pages 251–262, Bombay, India, 1996. Morgan Kaufmann.
9. M.V. Mannino and W. Effelsberg. Matching techniques in global schema design. In *Proc. of International Conference on Data Engineering (ICDE'84)*, pages 418–425, Los Angeles, California, USA, 1984. IEEE Computer Society.
10. E. Metais, J.N. Meunier, and G. Levreau. Database schema design: A perspective from natural language techniques to validation and view integration. In *Proc. of International Conference on Conceptual Modeling (ER'93)*, pages 190–205, Dallas (Texas), USA, 1993. Lecture Notes in Computer Science, Springer-Verlag.
11. L. Palopoli, L. Pontieri, G. Terracina, and D. Ursino. Intensional and extensional integration and abstraction of heterogeneous databases. *Data & Knowledge Engineering*, Forthcoming.
12. L. Palopoli, D. Saccà, G. Terracina, and D. Ursino. A unified graph-based framework for deriving nominal interscheme properties, type conflicts and object cluster similarities. In *Proc. of Fourth IFCIS Conference on Cooperative Information Systems (CoopIS'99)*, pages 34–45, Edinburgh, United Kingdom, 1999. IEEE Computer Society.
13. L. Palopoli, D. Saccà, and D. Ursino. An automatic technique for detecting type conflicts in database schemes. In *Proc. of ACM Conference on Information and Knowledge Management (CIKM'98)*, pages 306–313, Bethesda, Maryland, USA, 1998. ACM Press.
14. L. Palopoli, D. Saccà, and D. Ursino. Semi-automatic techniques for deriving interscheme properties from database schemes. *Data & Knowledge Engineering*, 30(4):239–273, 1999.
15. S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258–274, 1994.
16. J.D. Ullman. Information integration using logical views. In *Proc. of International Conference on Database Theory (ICDT'97)*, pages 19–40, Delphi, Greece, 1997. Lecture Notes in Computer Science, Springer-Verlag.
17. D. Ursino. Deriving type conflicts and object cluster similarities in database schemes by an automatic and semantic approach. In *Proc. of Symposium on Advances in Databases and Information Systems (ADBIS'99)*, pages 46–60, Maribor, Slovenia, 1999. Lecture Notes in Computer Science, Springer-Verlag.
18. J. Widom. Research problems in data warehousing. In *Proc. of International Conference on Information and Knowledge Management (CIKM'98)*, pages 25–30, Baltimore, Maryland, USA, 1995. ACM Press.
19. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

Constraint Satisfaction for Reconciling Heterogeneous Tree Databases

Hajime Kitakami, and Mitsuko Nishimoto

Hiroshima City University
3-4-1 Ozuka-Higashi, Asa-Minami-Ku,
Hiroshima 731-3194, JAPAN
E-mail: kitakami@its.hiroshima.cu.ac.jp

Abstract. In order to simplify the reconciliation of two heterogeneous tree databases, we must minimize the number of crossovers in a directed graph constructed using two subtrees selected from the databases. This paper proposes a method for minimizing the number of crossovers in the directed graph. To find the directed graph with the minimum number of crossovers, the method maintains zero-crossovers in each ordered subtree. The resulting directed graph is defined as a semi-optimal solution satisfying the zero-crossover constraint for edges connecting two leaf sequences. It is computed by changing the order of non-leaf nodes in each hierarchical level of the ordered tree and swapping leaf nodes in each of the two leaf layers. To maintain the zero-crossover constraint for each ordered tree in the matrix transformation, the method also finds the two leaf clusters that contain half of the leaf nodes and swaps the leaf clusters.

1. Introduction

The recent evolution of the Internet has facilitated access to a large amount of different information in the form of text, images, and so on. In many instances one might wish to integrate different databases available on the Internet. However, even when two different sets of information have similar contents, they almost never use the same semantics or syntax because they have been created in different areas or fields in a loosely coupled environment.

This paper develops a method to reconcile two heterogeneous tree databases into an understandable reconciliation tree and provides insight into the methodology. Goodman et al. [1] first introduced the concept of reconciliation in the field of molecular biology. Using their concept, it is important to develop mechanisms to search for the directed graph with the minimum number of crossovers. We define the directed graph constructed from two subtrees, which are retrieved from two heterogeneous databases, and propose a method for searching for the graph satisfying the minimum-crossover constraint, which is a semi-optimal solution for the zero-crossover constraint.

Computer reconciliation is useful for comparing heterogeneous gene trees, improving a taxonomic tree using a gene tree [2], and discovering conceptual

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 624–633, 2000.

differences among different people involving diverse structures [3]. Moreover, this method is concerned with retrieving data that can be represented by a tree structure [4]. A method for finding the directed graph with minimum-crossovers has applications in many fields and its development is very significant.

From another perspective, the search problem is simply related to constraint databases [5] derived from constraint programming. Typically, a query specifies satisfying the minimum-crossover constraint in the databases. Therefore, a method for satisfying the minimum-crossover constraint is not only useful for specific domains, but also for general domains in constraint databases.

This paper proposes an effective new method for finding the directed graph satisfying the minimum-crossover constraint, after defining an initial directed graph constructed using two subtrees retrieved from two heterogeneous tree databases.

The rest of this paper is organized in the following way. Section 2 introduces reconciliation work. Section 3 provides an overview of constraint databases. We propose a data model including the tree structure, the zero-crossover constraint, the interconnection matrix and basic operations in Section 4. The new method for satisfying the minimum-crossover constraint using the interconnection matrix is presented in Section 5 and an implementation in Prolog is discussed in Section 6. We summarize our results in Section 7.

2. Reconciliation work

Figure 1 shows an example of different tree structures [2] in the field of molecular biology. The species tree on the right side of the figure is constructed by classifying the morphological characteristics of biological species. In general, species trees can be up to 25 to 30 levels deep. Each green square in a leaf node represents a species name. Each of the blue and red circles represents the classification name of a species from the leaf level to the root level in the species tree. For example, the gorilla has five non-leaf nodes in this figure. The phylogenetic tree on the right is computed by comparing DNA sequences. Each leaf node represents a species name, and each non-leaf node represents a branch point in molecular evolution; the length of the branch represents the evolutionary distance. Both types of trees are different structures, and understanding biological diversity requires a comparison of both.

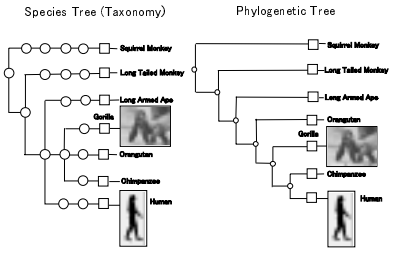


Fig. 1. An example of different structures

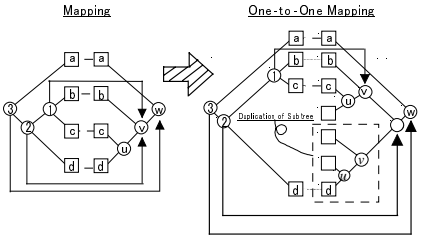


Fig. 2. An example of reconciliation

Figure 2 shows an example of reconciliation including the comparison of the different tree structures. Each side of the figure represents a graph constructed by two ordered subtrees with zero-crossover. Each graph represents a mapping from the left-hand tree to the right-hand tree. In the left-hand graph, the multiple nodes "1" and "2" in the domain map onto the same node, "V," in the range. Since the reconciliation needs to achieve one-to-one mapping, the subtree with the root "V" is duplicated, and the duplicated subtree is added to the original tree in the range. The domain does not map onto the white squares, and these are either extinct or uncollected in a biological sense.

3. System overview

Figure 3 shows the system configuration that we developed. Two heterogeneous tree databases, each with a set of edges (binary relationship), are included in the directed graph. The two ordered sets used to represent the databases each store ordering tuples in secondary memory. A typical constraint query for reconciliation work using the tree databases follows:

In the query, the search engine retrieves two ordered subtrees with a set of leaves given by the user from the tree databases and makes one ordered graph from the two ordered subtrees. Even if we make an ordered tree using this order, the two ordered subtrees do not have identical leaf nodes. Therefore, the constraint solver proceeds to change the order of the initial graph and finds that the two ordered subtrees satisfy the zero-crossover constraint while minimizing the number of crossovers between the two leaf layers. After that, the query processing system outputs the new graph defined as the two ordered subtrees found using the constraint solver.

Figure 4 shows an example of this situation. The constraint solver in Figure 3 searches the initial ordered subtrees retrieved from two heterogeneous tree databases for the two ordered subtrees with the minimum number of crossovers. In other words,

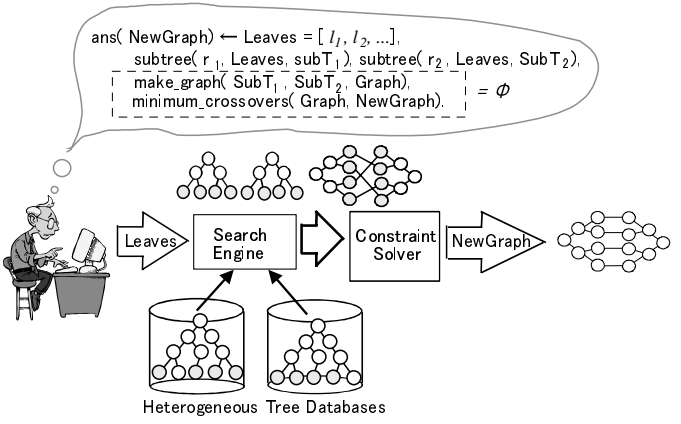


Fig. 3. System configuration

the constraint solver proceeds to change the order of the initial graph shown in Figure 4 and find the two ordered subtrees satisfying the zero-crossover constraint for each subtree while minimizing the number of crossovers between the two leaf layers. Figure 5 shows an example of the result. A computer display visualizes the two ordered subtrees found using the constraint solver.

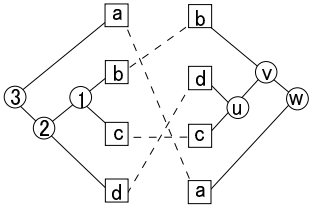


Fig. 4. An example of a directed graph representing two subtrees

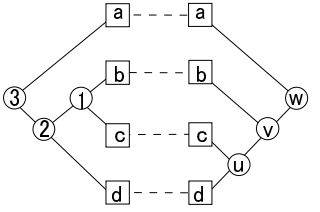


Fig. 5. An example of a directed graph with the minimum number of crossovers

4. Data Model

This section defines the data model used to represent both heterogeneous tree databases and the subtrees retrieved from the databases. We also define the order of each tree database used to store ordered tuples in secondary memory. This definition allows us to determine the order of each ordered subtree using the order of the tuples sent from the search engine. Moreover, we specify that two subtrees used to construct the directed graph satisfy the zero-crossover constraint after defining the interconnection matrix to represent the directed graph. Finally, we define the basic operations for the tree structure. The data model proposed in this section can be used to process advanced problems without satisfying the crossover constraint.

4.1 Tree Structure

Each tree represents one binary relationship with a tree model in each database. The tree model has known properties such as unique root node, no cycle and no nodes with no incoming edges other than the root.

For simplicity, we normalize all trees with a height of n-1. If the depth of a leaf is less than n-1, we add dummy nodes so that the path length from the root to the leaf is n-1. We call the directed graph, g(V, R, n, Σ), an ordered tree T with order Σ and height n-1, if the directed graph is an n-level hierarchy ($n \geq 2$) [6], [7] that satisfies the following conditions, where V is a set of nodes and R a set of edges:

(1) V is partitioned into n subsets. That is:

$$V = N_1 \cup N_2 \cup ... \cup N_n \qquad (N_i \cap N_j = \phi,\ 1 \leq i < j \leq n),$$

where N_i is called the i^{th} level and n is the number of levels in the hierarchy.

(2) N_1 contains one element, which is called the root of T.

(3) R is partitioned into n-1 subsets. That is:
$R = B_1 \cup B_2 \cup ... \cup B_{n-1}$ $(B_i \cap B_j = \phi, i \neq j)$, $B_i \subset N_i \times N_{i+1}$, $i \leq j \leq n-1$.
where any two edges, (p_1, q) and $(p_2, q) \in R$, satisfy $p_1 = p_2$. For every edge, $(p, q) \in R$, p is called the initial (or parent) node and q is called the final (or child) node.

(4) The set of nodes with zero in-degree consists of N_1 only. The set of nodes with zero out-degree is N_n and each node is called a leaf.

The order σ_i of N_i is given for each i, where the term 'order' means the sequence of all nodes of N_i; $\sigma_i = d^{(i)}_1, d^{(i)}_2, ..., d^{(i)}_\alpha$, where α denotes the number of nodes of N_i. The ordered binary relationship of the sequence is represented by the binary relationships, $d^{(i)}_1 <_B d^{(i)}_2$, $d^{(i)}_2 <_B d^{(i)}_3$, ..., $d^{(i)}_{\alpha-1} <_B d^{(i)}_\alpha$. The n-level hierarchy defined by the directed graph is denoted by $g(V, R, n, \Sigma)$, where $\Sigma = (\sigma_1, \sigma_2, ..., \sigma_n)$.

4.2 Zero-Crossover Constraint

Let us consider a bijective mapping $h : V_1 \rightarrow V_2$ between two trees, $T_1 = g(V_1, R_1, n_1, \Sigma_1)$ and $T_2 = g(V_2, R_2, n_2, \Sigma_2)$, where $(V_1 - Leaf_1) \cap (V_2 - Leaf_2) = \phi$. $Leaf_j$ is the set of leaves for T_j. We associate $Leaf_2$ with $Leaf_1$ by using the same names, but the two nodes with the same name are quite distinct from each other in the directed graph. When none of the branches in tree T_j crossover, this condition is called the zero-crossover constraint (CS_1). The association of $Leaf_2$ to $Leaf_1$ defines the edges connecting $Leaf_1$ to $Leaf_2$. When there are no crossovers between the edges, this is the zero-crossover constraint for both T_1 and T_2 (CS_2). These two constraints are generally called zero-crossover constraints.

4.3 Interconnection Matrix

In general, the graph constructed by the two ordered subtrees does not have the zero-crossovers shown in Figure 4. This section relates to minimizing the number of crossovers between the two leaf layers. Of course, the crossover of each ordered tree should remain zero in the graph.

In order to solve the minimum-crossover problem, we define an interconnection matrix representing the two leaf layers shown in Figure 6. There are two ordered leaf lists. The ordered leaf list for the left-hand tree is represented as "a, b, c, d". The right-hand tree is "b, d, c, a". Leaf "b" in the right-hand tree and the ordered leaves "a, b, c, d" in the left-hand tree have a connective relationship that is defined as the vector (0 1 0 0) located in the first row of the matrix. Since leaf "b" is connected to the second leaf in the left-hand ordered tree, the second element in the vector is unity. In general, the number of crossovers between the leaf lists is defined in the following manner:

$$\text{Crossovers} = \Sigma\, m_{j,\beta}\, m_{k,\alpha}\ [1=j<k=n,\ 1=\alpha<\beta=n] \qquad (\mathbf{1})$$

The equation calculates that there are 4 crossovers for the matrix in the figure. We try to minimize the number of crossovers by swapping any two leaves for each ordered

tree. When the number of crossovers is zero, the connection matrix is a unit matrix, as shown in the lower figure.

Figure 7 shows the usefulness of two swap operations in the construction of a unit matrix from a non-unit matrix. Let us consider moving the white circle to the pivot position represented by the white square. We assume that we didn't move any two marks with a cross to the pivot position. First, we need to swap the 4th pivot row with the 6th row that contains unity. Second, we need to swap the 4th pivot column with the 6th column that contains unity.

4.4 Basic Operation

To satisfy the zero-crossover constraint, we propose two basic operations. The first type is the breadth-first search, which is useful in constructing the ordered tree satisfying the zero-crossover constraint. The second type is the search of the two maximal subtrees, which is useful in distinguishing the two leaves. The set of leaves with respect to the subtree is called a cluster.

The breadth-first search to construct the initial ordered tree from the database is defined as follows. Let us consider a ordered tree T= g (V, R, n, Σ). A forward search for R and S, denoted as R∇S, is defined as {(b, a) | there exists at least one value of variable (c, b) such that (b, a) $\in$ R and (c, b) $\in$ S}, where S is a subset of R. The tree T = g(V, R, n) has known properties, such as a unique root node, no cycles, and no nodes with no incoming edges other than the root. The results of the forward search have the order of the tuples send from the search. Let $S \subseteq R$, $T_1 = S$, $T_i = R\nabla T_{i-1}$, $2 \le i \le n$, $T_n = \phi$. The search for R and S is defined as $R^* = \cup_{i>0} T_i$. This is a recursive forward search [8]. The following algorithm represents the search:

```
R* := S;  R' := S;

while(R' ≠ϕ) { R' := R∇R' ; R* := R*∪R'; }

Result:  = R*;
```

If the subset S is equal to the set of edges with the initial node of the root, we can obtain the all edges of the tree in set Z^*. In this process, the order of the edges with the same initial node is naturally determined by the original sequence of the tuples

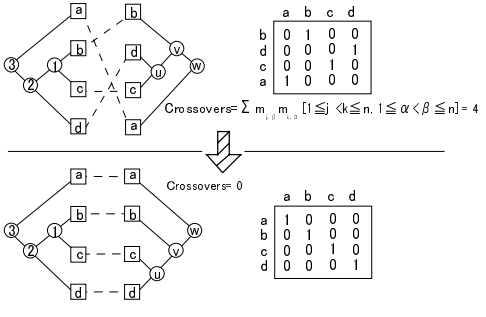


Fig. 6. An example of interconnection matrix

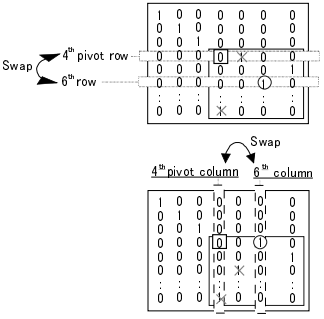


Fig. 7. An example of swap operations

stored in the database. When the process finishes, we obtain the initial ordered tree T = g(V, R, n, Σ) satisfying the zero-crossover constraint. In the opposite way, we can easily define a recursive backward search.

Let us consider the method used to search the two clusters for the ordered tree. The two clusters are useful in distinguishing two different leaves, d, c ∈ N_n, where d≠c. The subtrees used to find the two clusters, C_1 and C_2, shown in Figure 8 can be searched using the following procedure:

(1) After finding the branch node, r, for two different leaves using the recursive backward search, d, c ∈ N_n, we select the two subtrees, $subT_1$ and $subT_2$, from all the subtrees connected to the branch node. When $subT_1$ and $subT_2$ include the respective leaves, d and c, the same nodes for $subT_1$ and $subT_2$ do not exist. $SubT_1$ and $subT_2$ are also the maximal trees selected from the possible subtrees to distinguish between two different leaves, d, c ∈ N_n.

(2) The two sets, C_1 and C_2, related to the leaves are computed from the two subtrees, $subT_1$ and $subT_2$, respectively, using the recursive forward search, where $C_1 \cap C_2 = \phi$.

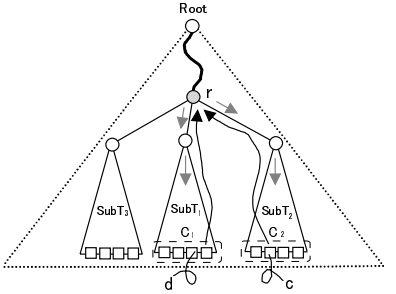


Fig.8. An example of a search of two leaf clusters in the database

5. Constraint Solver

This section proposes a method for minimizing crossovers in the directed graph constructed from two subtrees, using the constraint solver shown in Figure 1. In order to minimize the crossovers in the directed graph, it is important that each ordered tree satisfy the first constraint (CS_1) and that the connections between the two leaf sequences optimize the second constraint (CS_2), which minimizes the crossovers in the graph. If we can satisfy CS_2 perfectly, there are no crossovers. Before optimizing CS_2, we must construct two ordered subtrees that satisfy CS_1. First, two ordered subtrees are constructed using the breadth-first search (recursive backward search) introduced in Section 4.4. Initially, there are crossovers in the leaf layers of the two trees, as shown by the dotted lines in Figure 4. It is important to reduce the number of crossovers without generating new crossovers within the branches of either tree. To do this, we propose a method for optimizing CS_2 using matrix transformations consisting of restricted swap operations between either rows or columns of an interconnection matrix. Matrix transformations that do not violate CS_1 are guaranteed by the new swap operations involving the previously stated clusters. CS_2 is optimized as the matrix transformations approach a unit matrix in which the value of the diagonal elements is unity and the other elements are zero. There are powerful algorithms [6], [7] using n-level hierarchies defined as directed graphs that can resolve this problem, but these algorithms are too general to resolve our specific

problem. Our algorithm is concerned with the interconnection matrix that represents 2-level hierarchies, which are defined as two leaf layers across two trees.

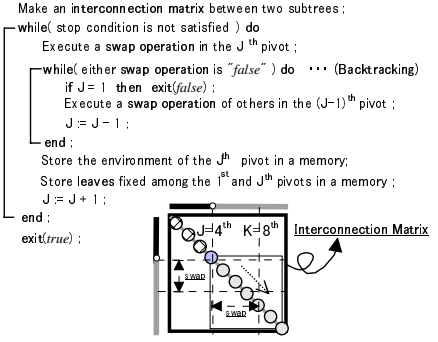


Fig. 9. Reducing Crossovers

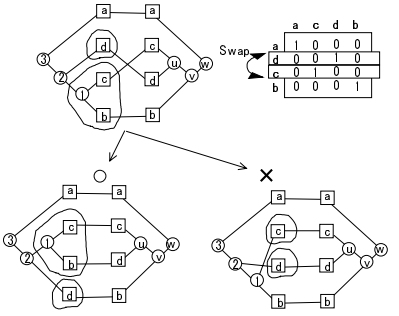


Fig. 10. An example of the new swap operation

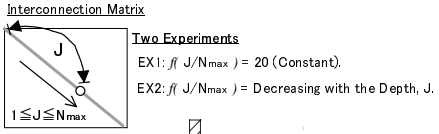
5.1 Reducing Crossovers

Figure 9 shows the algorithm for reducing the number of crossovers and includes the previously stated swap operations. After making an interconnection matrix between the two subtrees, the swap operation in the outer loop is carried out from the upper-left to the lower-right corner of the matrix along the diagonal elements. Swap operations at the $(J-1)^{th}$ pivot in the inner loop are related to backtracking if either swap operation in the outer loop is rejected at the J^{th} pivot. When one of three stop conditions is satisfied, the algorithm exits the outer loop, and the algorithm outputs a semi-optimal graph with respect to the minimum crossovers.

Either of the previous swap operations shown in Figure 7 generates new crossovers within the branches of either tree in the graph. The lower-right graph identified by a cross has the new crossovers in the left tree of Figure 10. This was caused by the previous swap operation, which exchanged two leaves in the tree. In order to avoid the generation of a tree without zero-crossover, we propose a new swap operation that exchanges two clusters in the tree. The lower-left graph identified by a circle is computed by the new swap operations.

5.2 Stop Conditions

There are three stop conditions of the previously stated algorithm. If each of them is satisfied, the algorithm stops the processing. The first stop condition relates to the pivot number. The second one relates to the number of crossovers. The last one is the most important condition in the presentation. It relates to the number of occurrences for a current semi-optimal solution. When the occurrences are greater than the value of the function, the algorithm stops the processing. The value of the function will be shown in detail in the next section.



J/Nmax	0.0-0.20	0.21-0.50	0.51-0.70	0.71-0.90	0.91-1.00
f(J/Nmax)	50	30	20	10	5

Fig.11. An example of stop condition

Table 1. Performance evaluation

Data	Number of Nodes	Minimum Crossovers	EX1: Constant		EX2: Variable	
			CPU Time (sec)	Final Crossovers	CPU Time (sec)	Final Crossovers
Case-1	25	0	0.1	0	0.1	0
Case-2	20	0	0.2	0	0.2	0
Case-3	9	1	0.3	1	0.2	1
Case-4	39	14	16.7	33	9.4	33
Case-5	117	32	26.2	32	19.2	32
Case-6	241	10	168.6	88	76.1	10

6. Implementation

The search engine shown in Figure 1 is implemented as a stored procedure in a relational database system, SYBASE. Moreover, we implemented a prototype of the constraint solver system shown in the figure in Prolog. Two heterogeneous databases were copied from the original databases available over the Internet using a Web browser in the relational database system. This section evaluates the system using two kinds of databases, a taxonomy database constructed by the authors [9] and a phylogenetic database that is part of the Jungle Project [10].

Figure 11 shows the previously stated stop condition related to the occurrences in detail. If the number of occurrences for the current semi-optimal solution in the processing is greater than the value of a function with the parameters "J" and "N_{max}," the stop condition is satisfied. We have experimented with two methods:
The first one is evaluated on the condition that the function is constant in any pivot position. The second one is that the function decreases with the depth, "J".

Table 1 shows the comparison of both methods. We evaluated six examples given by molecular biologists. The examples shown in the table minimize the crossovers between species and phylogenetic trees. The performance is evaluated on the condition that the number of nodes for the species trees is between 9 and 241. The result was that the second method was faster than the first one.

7. Summary

We have proposed a constraint solver with a new swap operation to try to find a semi-optimal solution. The constraint solver minimizes the number of crossovers for a directed graph constructed from ordered subtrees received from two heterogeneous tree databases.

The interconnection matrix defined between two leaf layers proved useful in the search of a solution. We have developed a powerful method to search the diagonal matrix by finding two clusters in the new swap operations. Moreover, the second stop

condition depending on the depth of the pivot position was more effective than the first stop condition. We envision the following future projects:

(1) Developing an extended reconciliation method, to solve more general computational problems.
(2) Developing intelligent tools to graphically modify the graph to support reconciliation work.

Acknowledgments

We thank Associate Professor Naruya Saitou and Dr. Satoshi Ota of the National Institute of Genetics for their information on biological issues, and the staff of the international DNA data banks for their help in obtaining the taxonomy databases. We also thank Dr. Akira Sato and Mr. Yasuma Mori of the Hiroshima City University for their helpful comments and for their assistance in preparing computer environment.

This work was supported in part by a Grant-in-Aid of Scientific Research (C) from Japan Society for the Promotion of Science, and a Hiroshima City University Grant for Special Academic Research.

References

1. Goodman, M., Czelusniak, J., Romero-Herrera, A. E., and Matsuda, G.: Fitting the Gene Lineage into its Species Lineage: A parsimony strategy illustrated by Cladograms Constructed from Globin Sequences, Systematic Zoology, Vol. 28 (1979) 132-168.
2. Page, R.D.M., and Charleston, M.A.: Reconciled Trees and Incongruent Gene and Species Trees, In: B Mirkin, F R McMorris, F S Roberts and A Rzhetsky (eds), Mathematical Hierarchies in Biology, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Vol. 37 (1997) 57-70.
3. Yoshida, T., Kondo, T., and Nishida, S.: Discovering Conceptual Differences among Different People via Diverse Structures, Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining, Beijing in China (1999).
4. De Marsicoi, M., et al.: Indexing Pictorial Documents by their Content, A Survey of Current Techniques, Image and Vision Computing, Vol. 15 (1997) 119-141.
5. Kanellakis P.: Constraint Programming and Database Languages: A Tutorial, Proc. of ACM POS'95 (1995) 46-53.
6. Warfield John N.: Crossing Theory and Hierarchy Mapping, IEEE Transaction on Systems, Man, and Cybernetics (1977) 505-523.
7. Sugiyama K., Tanaka S., and Toda M.: Methods for Visual Understanding of Hierarchical System Structures, IEEE Transaction on Systems, Man, and Cybernetics (1981) 109-125.
8. Ullman Jeffery D.: Database and Knowledge-Base Systems, Vol.1, Computer Science Press (1988).
9. Kitakami Hajime, Mori Yasuma, Arikawa Masatoshi, Sato Akira.: Integration Method for Biological Taxonomy Databases in the Presence of Semantic Heterogeneity, IEICE, Vol. J-82-D1, No.1 (1999) 303-314.
10. Saitou Naruya et al.: Phylogenetic Tree Database (JUNGLE) , http://smiler.lab.nig.ac.jp/jungle/jungle.html (1998).

A Mobile Agent Carrier Environment for Mobile Information Retrieval

T. I. Wang[1]

Laboratory of Intelligent Network Applications
Department of Engineering Science
National Cheng Kung University
Taiwan
{wti535}@mail.ncku.edu.tw

Abstract. This paper describes the use of mobile agent technologies in building a framework for the mobile information retrieval systems. A Mobile Agent Carrier Environment-MACE[1] is first introduced. It is designed in allusion to a new computation notion called the service-on-demand. A specially designed protocol, the Service Protocol, provides an easy and flexible way of building service based applications. Based on the MACE, a framework is then established for mobile information systems. The framework supports remote information retrievals for mobile platforms, such as laptop computers or PDAs. It uses both fixed and mobile agents for carrying out information access in a variety of ways. Specific service stations are also included in the framework to cope with the difficulties arose from the unique characteristics of wireless networks.

1 Introduction

Mobile agents are recognized as software modules that move from host to host. They may interact with each other and access distributed resources in a heterogeneous network [KK1]. Mobile agents consume fewer network resources and, therefore, are particularly useful for developing distributed applications. There have been studies, implementations, and prototype applications of mobile agents [CTB1], [KGN+1], [KLO2]. Most of them use a model combing client-server computation and mobile code technology, and are for computers with permanent network connection.

Meanwhile, mobile information systems have been attracting more and more researchers as they discover the benefits of being able to connect to distributed information resources without any spatial and temporal constrain[Gr1]. However, mobile information retrievers such as PDAs or laptops also raise new issues in the context of communication. Drawbacks like low bandwidth and high latency of a present wireless network make a user tend to connect a mobile platform to a wireless network for a period as short as possible. Another well-marked problem

[1] The research was fully supported by the National Science Council, Taiwan, under the project NSC89-2213-E-006-056

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 634–643, 2000.

is that a mobile platform may get a different network address assignment each time it connects to a wireless network. This makes a mobile platform more appropriate as an information sender than as a receiver if it keeps moving. Even though these problems can be solved in the near future, there are still mobility-related difficulties to be eliminated. As we will see, autonomous mobile agent is one of the promising technologies to respond to these questions.

In this paper, a Mobile Agent Carrier Environment-MACE is first introduced. Its design supports mobile computing primitively. Based on the MACE, a framework is then established for mobile information systems. In section 2, the design philosophy behind the MACE is advocated. This philosophy gives the reasons for designing a carrier system instead of a complete mobile agent developing system. Section 3 describes the details of the MACE, including the functionality of each component. Section 4 presents the framework that supports a mobile information system. In section 5, comparisons between MACE and other mobile agent platforms are made. Finally, in section 6, conclusion is made and possible future works are described.

2 Trends and The MACE Approach

In contrast to the client/server model in distributed computing paradigm, which ships data to remote application codes, the computation model of mobile agent paradigm migrates application codes to remote data. The motivation is to reduce network traffic by reducing interactions and data transfers between distributed components. The aim is to gain more efficient bandwidth utilization and higher system availability. This code mobility greatly benefits those distributed applications that compute with simple logic and huge remote data.

Along with the advance of network technology a new concept of computation, referred to as service-on-demand, is also gradually taking its shape. The core of it is the service base. A service base resides in a service station and is a collection of various services supplied by different service providers. Users may choose their necessary and favorite services in a base, and pay for the execution of them afterwards. The same scenario has been happening in our daily life. Customers are using yellow page service, consultant service etc. and there are so many such service providers.

Mobile agents are inherently suitable for seamlessly realizing such a notion, and for easily building service oriented applications. At the first impression, it might seem naturally to implement services as mobile agents, and agents migrate to the site from which requests are made. After a second thought, it turns out that implementing a service as a mobile agent might not be meaningful in such situations when a computation happens at the same site on which the data it consumes resides. Unfortunately, most of the *service-on-demand* applications belong to this category. So what's the point in using mobile agents to build such kind of systems?

Perhaps a very good answer to the question is to support the so-called "fire-and-forget" functionality. Users, after launching the agents, are free from engag-

ing in any interactions with the code that actually performing computations. They just collect the results later in a convenient time and place. For example, before taking a teatime break, a user can launch agents to visit air flight and ticket information (service) sites at three airline companies. And, after the break, without being attended, the agents would have chosen a ticket of right time and good value for the user.

While from another point of view, giving an entire mobile agent developing system to a user seems inadequate in the *service-on-demand* application domain. To provide a service, the service provider should not expect a user to code an agent to interact with the server at the service station. What a service provider has to do is to supply the user with an attractive user interface to collect necessary arguments for instantiating the service. The business logic should be left to the service provider itself. In such cases, it is the list of arguments that will instantiate the service has to migrate, not the entire user interface. This is an important philosophy behind the design of **Service Protocol** in the **MACE** system.

2.1 The MACE Approach - The Service Protocol

Though targeted at *service-on-demand* applications, MACE's design retains the flexibility of a general mobile agent system. A **Service Protocol** is established to realize the *service-on-demand* concept, in which distributed resource access and management are abstracted as services. In the Service Protocol, a service is fulfilled by executing two closely related components, an **Agentlet**, and a **Serverlet**. In general, both of them are developed by a same service provider. The Agentlet is distributed to some dedicated servers that provide directory services to all the users. The Serverlet is stored at a service station where the service will be actually carried out. Agentlets of frequently used services are downloaded and cached locally in a user's local directory. Along with the advance of network technology a new concept of computation, referred to as service-on-demand, is also gradually taking its shape. The core of it is the service base.Users invoke Agentlets via an Agent Creator. An Agentlet, after invoked, may produce one or more service items, each of which corresponds to an invocation of a Serverlet at a service station. Thus, an agent carries service items instead of codes, and a service item is the connection between an Agentlet and a Serverlet.

The beauty of the service item is that its content can range from as simple as a list of arguments to as complex as a segment of code written in a customer-defined scripting language. In the latter case, the associated Serverlet of such a service item becomes an interpreter, and the MACE is in reality as well in name a **Mobile Agent Carrier Environment.** Therefore, the Service Protocol is simple yet powerful, confining mainly the rules for transferring information among three parties - Agentlet, Agent Creator, and Serverlet.

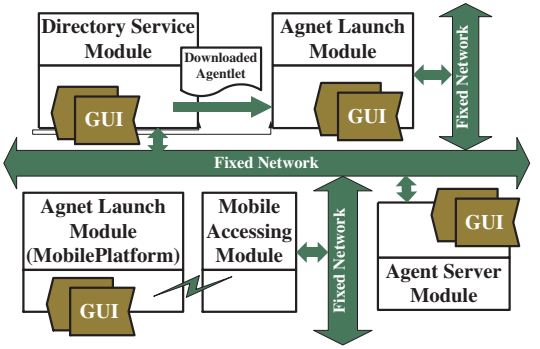


Fig. 1. The MACE System

3 MACE: the Mobile Agent Carrier Environment

The whole MACE system is divided into four major parts as shown in Figure 1. The **Agent Launch Module**, as the name suggests, creates and launches agents into the network. Launched agents will roam through the network and reach some **Agent Server Module** where services are carried out. Agent Launch Module is the implementation of **Agent Creator** in the Service Protocol. While creating agents, users can consult the **Directory Service Module** to find out preferable or favorite services and download the associated agentlets. The **Mobile Accessing Module** is a bridge for mobile platforms to launch and collect agents to and from the network.

3.1 Directory Service Module

The Directory Service Module, as its name suggests, helps a user to find specific services. It is a part of the implementation of Service Protocol in MACE. Agentlets of a variety of services are managed and entries for these services are kept in the Service Directory by the Directory Server in the module. The Service Protocol defines how a service provider can register a certain kind of services with the Service Directory. In the mean time, associated agentlets of these recorded services must be uploaded to the service station. A service provider can renew the agentlet of a service when it is necessary. Agentlet version information is maintained by the Directory Server automatically.

3.2 Agent Launch Module

There are two types of Agent Launch Module, one is for hosts with permanent connection to the network, and the other is for mobile platforms. They have almost the same components except the physical links to the permanent network. The main functionality of this module can be shown as in Figure 2.

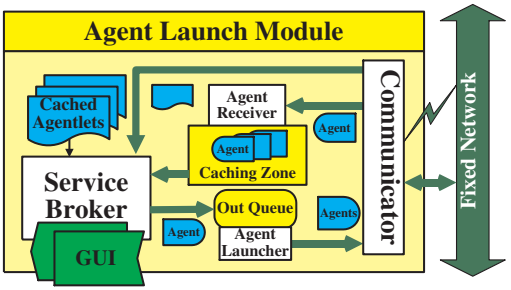


Fig. 2. Functionality of Agent Launch Module

The **Service Broker** displays, via a GUI, to the user the services whose associated agentlets are cached locally. Alternatively, when requested, it may consult remotely a specific Directory Service Module for a wider range of services. The associated agentlet of a service that is found in the remote Directory Service Module is downloaded if the user means to use that service. The associated agentlet is then invoked first to collect from the user the necessary information for carrying out the service. For instance, this information may be the key attribute of a distributed database table to be sorted, or it may be the business logic of a transaction for some electronic commerce. Several agentlets may be activated in the course of creating an agent. In other words, many service items may be packed into a single agent, and each of them with their own service information.

A service item normally starts with a migration command followed by some services. In addition to the service items, every agent also contains an ID field, which uniquely identifies the agent and the user who creates the agent. After an agent is created, it is launched by the Agent Launcher into the network to start its journeys. For mobile platforms, there is a Temporarily Waiting Lounge. Agents created in a mobile platform will stay in that lounge until a proper connection to the permanent network is made. An agent, after roaming through the network and having all its service items served, does not necessarily return to it birthplace. Users can specify the final location for an agent to stop. This capability allows a mobile platform to release agents to work and later collect them at another place, a great support for mobile computing. In either situation, agents are received and put into a Caching Zone by the Agent Receiver. Users can then inspect the results of all the services carried by an agent via the Service Broker. Other housekeeping works, such as terminating an agent and saving the results, are also the responsibility of the Agent Broker.

3.3 Agent Server Module

The Agent Server Module plays the major role in serving the service items in a received agent. It resides at a service station that provides different kind of services. Figure 3 shows the functionality and components of an Agent Server Module. The Agent Fetcher/Interpreter fetches an incoming agent, picks out the

service items targeted for this service station, and extracts service information from these items. These extracted service information is then handed to the Serverlet Invoker to initiate and instantiate the associated serverlet. When the service is finished, the execution result is packed into the original agent with some indexing information. Finally, the Agent Launcher fires the agent into the network to continue its unfinished journey.

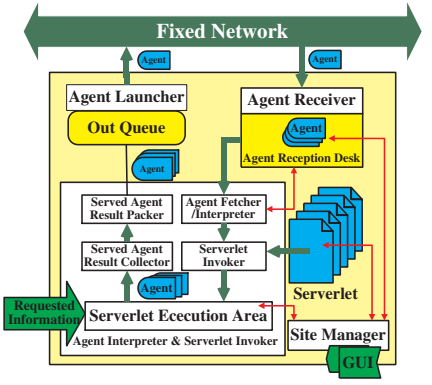


Fig. 3. Functionality of Agent Server Module

3.4 Mobile Accessing Module

To support mobile computing, MACE introduces a component named the Mobile Accessing Module. In addition to the operations of normal wireless network connections, this module supports the so-called *disconnected operations* for mobile platforms [Gr1]. The functionality of the module is shown as in Figure 4.

The Mobile Accessing Module resides in a docking host that operates a physical device to accept wireless connection from mobile platforms. The Accessing Point is a bridge between wireless and permanent network. Agents created in a mobile platform wait in the Temporarily Waiting Lounge there until a proper wireless connection is established between the mobile platform and the docking host. They then quickly jump off to the docking host, from which they are launched into the permanent network to carry out their own missions. This allows the mobile platform to disconnect from the network as soon as possible, and to stay disconnected while the computation is going on. On the other hand, mobile agents, after being properly served, may return to the original or a pre-specified docking host to find that the mobile platform is still disconnected. They then enter another Temporarily Waiting Lounge to wait for the reconnection. Once the reconnection happens, mobile agents will quickly jump back to the mobile platform and report their achievements.

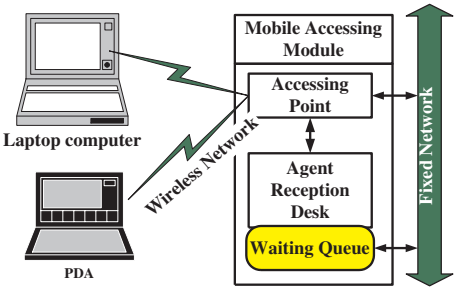


Fig. 4. Functionality of Mobile Accessing Module

4 A Framework for Mobile Information systems

MACE in itself is effective enough to be the infrastructure of a distributed information system. However, to build a framework out of it for effective on-the-go information access, two additional components must be added. One is a **Fixed Agent** that interacts with the local information processing system such as the DBMS; the other is a **Staging Post** that serves as a temporary stopping place for mobile agents with retrieved data. The former is built as a special persistent Serverlet, and the latter is adapted from a docking host. The resulting framework is shown as in Figure 5.

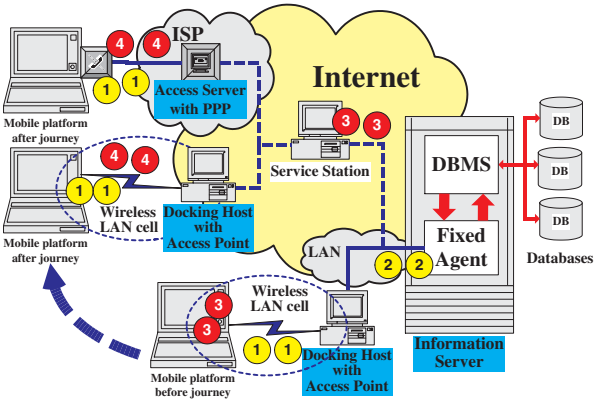


Fig. 5. The framework

4.1 The fixed agent

A fixed agent resides on a specific Information Server and is persistently running. It interacts with the information retrieval system in the server to access the

data needed by an incoming mobile agent. For instance, it gives to a Data Base Management System a set of SQL queries handed in by a mobile agent, and later collects the result tuples from the DBMS. Before handed over to a mobile agent, the retrieved information is divided by the fixed agent into a set of data packets of specific format. Mobile agents are responsible for carrying these data packets to the mobile platform that initiates the data retrieval, wherever it may be.

To acquire maximum flexibility, three different but related modes for retrieving information are supported in the framework.

1. Before leaving for some remote destination, a user may make a local access from a mobile platform to the information server for some desired data. Mobile agents are released to interact with the fixed agent in this mode (①and②in Figure 5). After the retrieval, all the data are kept in these mobile agents, which now reside in the user's mobile platform (③in the mobile platform before journey in Figure 5). The user can now leave for the destination. This is just like an ordinary access to the information base at a desktop, except there is a docking host acting as a bridge between the information server and the mobile platform.
2. A user may, as in the mode above, send out mobile agents to collect information, except that the user can disconnect from the docking host right away and leave for some remote destination. In this way, information-processing jobs that are time consuming can be done while the user is travelling. The staging post for these mobile agents must be pre-determined. Usually, it is a staging post near a user's final destination (③in the Staging Post in Figure 5). At that location, the user may gather up these mobile agents later (④in the mobile platform after journey in Figure 5).
3. While at a remote location, for any reason, a user may need to make an access to the information server. Mobile agents are again sent back home to interact with the fixed agents to collect the desired information(①in the mobile platform after journey in Figure 5). They may go back directly to the mobile platform if the user decides to stay connected and wait; or they may, as in mode 2, stop at a staging post if the user decides to disconnect temporarily from the network. The user later will gather up these mobile agents to receive the information(④in the mobile platform after journey in Figure 5).

As can be seen, whichever mode a user may use, the released mobile agents and the fixed agent will interact with a same pattern. This multi-mode design not only simplifies the complexity but also maximizes the flexibility of the framework.

4.2 The Staging Post

A Staging Post is a simplified version of a docking host. It contains a module similar to the Mobile Accessing Module, except that the wireless Accessing Point component is replaced by a managing process that uses a permanent link for network connection. The managing process is responsible for caching and

remanding mobile agents. Mobile agents, after received, are cached in the Agent Reception Desk, waiting for their owner to collect. Once a user is certified to connect to a Staging Post, mobile agents that are on her/his behalf will be sent back to his mobile platform. It may seem odd not to use a docking host directly. The reason behind this is that a user may not always have access to a docking host that is equipped with MACE. Even worse, the user may have to connect to the network by dialing to an Internet (or Network) Service Provider. Under such circumstances, a nearby Staging Post will provide normal services for the user.

5 Related Work

Many prototype or commercialized Mobile agent systems have been proposed and developed. A well-known mobile agent system is Telescript developed at General Magic [Wh1]. Telescript uses a proprietary script language to create agents. It also supports mobile platforms and has been successfully used on Personal Digital Assistants (PDA). Another script-based mobile agent system is Agent Tcl [Gr1], [KGN+1], developed at Dartmouth College. A set of special commands was added to an existing high-level scripting language Tcl, developed in 1987, to create Agent Tcl. An agent uses these commands to migrate from host to host and to communicate with other agents. Like MACE, these two systems use a proprietary script language to create passive agents. But, unlike MACE, users of these systems have to create these agents by themselves.

Among JAVA-based systems, Aglets, [Ag1], developed at IBM's Tokyo Research Laboratory, has been very popular for some time. It is praised for its GUI that makes the world of agents accessible to the JAVA novice. Another JAVA-based mobile agent system is Odyssey [Ge1], also by General Magic. Though implemented purely in JAVA, it still incorporates some of the concepts previously developed for Telescript. One unique feature of the Odyssey system is its audit trail mechanism to help programmers debugging their agents.

However easy theses systems may claim in creating agents, users still have to programming the agent, the service, or even the business logic by themselves, not to mention the tedious debugging work in a complex system in which agents interact heavily. MACE eliminates this by using the Service Protocol and the higher-level customer-defined script languages. Users are free from programming, and service provides are freely to use their own language for developing Agentlet and Serverlet as long as they are confined to the protocol. Maybe the only limitation is that service items generated by an Agentlet should be able to be put into an agent carrier (by buying a ticket?).

6 Conclusion and Future Work

This paper has presented the design and the implementation of a Mobile Agent Carrier Environment-MACE for distributed computations and information retrieval. A simple yet powerful Service Protocol is described, which makes service

providers free from being confined to a developing environment. On the other hand, using MACE and its underlying developing tool, one can still develop an entire sophisticated distributed system.

An agent tracking mechanism is being planned for the MACE. It will be used to trace the existence and the location of all created mobile agents in order to direct the dangling agents back to the host from which they were launched. A coordinator, which masterminds the cooperation between agents, is also under investigation. This will include a GUI for MACE users to precisely specify the relationship between service items and between agents. For the time being, MACE is targeted at applications in which agents execute to produce only simple short results and agents carry these results while they are travelling. For those applications that might transfer voluminous results between distributed sites, a logistics and delivery system is being investigated.

References

[Ag1] Aglets: Mobile Java Agents, IBM Tokyo Research Lab, URL = http: *//www.ibm.co.jp/trl/projects/aglets*

[CTB1] S. Covaci,; Zhang Tianning; I. Busse, "Java-based intelligent mobile agents for open system management", In Proceedings of Ninth IEEE International Conference on Tools with Artificial Intelligence, Page(s): 492 -501, 1997.

[FGS1] W.M. Farmer, J.D. Guttman and V. Swarup, " Security for Mobile Agents: Issues and Requirements, ". Proceedings 19th National System Security Conf. (NISSC 96), 1996, pp.591-597.1997.

[GK1] Robert Gray, David Kotz, Saurab Nog, Daniela Rus and Geoge Cybenko. "Mobile Agents for mobile computing.", Technical Report PCS-TR96-285, Department of Computer Science, Dartmouth College, May 1996.

[Gr1] R.S. Gray. "Agent Tcl: A Flexible and Secure Mobile-Agent System," in Proc. Fourth Annual Tcl/Tk Workshop (TCL 96), 1996.

[Ge1] General Magic, "Agent Technology: General Magic's Odyssey", *http : //www.genmagic.com/html/agent*$_o$*verview.html*, 1997.

[KLO1] G. Karjoth; D. B. Lange; M. Oshima, "A security model for Aglets", IEEE Internet Computing, Volume: 1 4 , Page(s): 68 -77, July-Aug. 1997.

[KLO2] Gunter Karjoth, Danny B. Lange, and Mitsutu Oshima, "A Security Model for Aglets,", IEEE Internet Computing, JULY,AUGUST 1997.

[KK1] Keith D. Kotay and David Kotz, "Transportable Agents ". In Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management, Gaithersburg, Maryland, December 1994

[KGN+1] D. Kotz; R. Gray; S. Nog; D. Rus; S. Chawla; G. Cybenko, "AGENT TCL: targeting the needs of mobile computers", IEEE Internet Computing Volume: 1 4 , Page(s): 58 -67, 1997.

[LO1] D.B. Lange and M. Oshima, Java Agent API: Programming and Deploying Aglets with Java, published by Addison-Wesley, Fall 1997

[Or1] J.J. Ordille, "When Agents Roam, Who Can You Trust?" Proc. Fitst Conf. on Emerging Technologies and Applications in Communications(etaCOM), May 1996.

[Wh1] J. E. White, "Telescript technology: The foundation for the electronic marketplace." General Magic White Paper, 1994.

A Skew-Insensitive Algorithm for Join and Multi-join Operations on Shared Nothing Machines

M. Bamha and G. Hains
{bamha,ghains}@lifo.univ-orleans.fr

LIFO, Université d'Orléans, B.P. 6759, 45067 Orléans Cedex 2, France.

Abstract. Join is an expensive and frequently used operation whose parallelization is highly desirable. However effectiveness of parallel joins depends on the ability to evenly divide load among processors. Data skew can have a disastrous effect on performance. Although many skew-handling algorithms have been proposed they remain generally inefficient in the case of multi-joins due to join product skew, costly and unnecessary redistribution and communication costs. A parallel join algorithm called fa_join has been introduced in an earlier paper with deterministic and near-perfect balancing properties. Despite its advantages, fa_join is sensitive to the correlation of the attribute value distributions in both relations. We present here an improved version of the algorithm called Sfa_join with a symmetric treatment of both relations. Its predictably low join-product and attribute-value skew makes it suitable for repeated use in multi-join operations. Its performance is analyzed theoretically and experimentally, to confirm its linear speed-up and its superiority over fa_join.

1 Introduction

Research has shown that the join operation is parallelizable with near-linear speed-up on shared nothing machines only under ideal balancing conditions. Data skew can have a disastrous effect on performance [1, 3, 13, 9, 4, 11, 5, 10].

Many algorithms have been proposed to handle data skew for a simple join operation, but little is known for the case of complex queries leading to multi-joins [8, 4, 7, 11]. In particular, the performance of PDBMS has generally been estimated on queries involving one or two join operations only [15]. However the problem of data skew is more acute with multi-joins because the imbalance of intermediate results is unknown during static query optimization [8]. Join algorithms described in previous works induce costly and unnecessary communications. In this paper we show that the communications involved in computing the join $R \bowtie S$ of two relations R and S can be reduced to a minimum by:

- an efficient computation of the semi-joins $R \ltimes S$ and $S \ltimes R$. This reduces the redistribution cost to only the tuples of relations R and S which will effectively be present in the join result.
- a dynamic choice of the build[1] and probe[2] relation for each bucket (sub-relation) of the two relations allowing to reduce the sub-set replication costs while avoiding attribute and join product skews.

[1] Build relation : the relation used to create the hash table.
[2] Probe relation : the relation used to probe the hash table.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 644–653, 2000.

Our present contributions to parallel relational query processing are:
1. A new parallel join algorithm called *sfa_join* (symmetric frequency adaptive join algorithm) for shared nothing machines. This algorithm has near-perfect balancing properties and supports flexible control of communications induced by intra-transaction parallelism. The sfa_join algorithm is based on:
(a) a symmetric partial duplication of data. (b) an improved version of the redistribution algorithm of *fa-join* [3, 1] which efficiently avoids the problem of attribute value- and join product skews.
2. An analysis of sfa_join in the scalable and portable BSP cost model. It predicts a negligible join product skew and a near linear speed-up for our algorithm, independently of the data and of the (shared nothing) architecture's bandwidth, latency and number of processors. This prediction is confirmed by a series of tests. Some of the tests also confirm the superiority of sfa_join over fa_join. Its suitability for multi-joins follows from the output's near-perfect balancing *at no extra processing cost.*

2 Load balancing in parallel joins and the BSP model

Parallel join usually proceeds in two phases: a redistribution phase by join attribute hashing and then sequential join of local fragments. Many such algorithms have been proposed. The principal ones are: *Sort-merge join*, *Simple-hash join*, *Grace-hash join* and *Hybrid-hash join* [12]. All of them (called hashing algorithms) are based on hashing functions which redistribute relations so that tuples having the same attribute value are forwarded to the same node. Local joins are then computed and their union is the output relation. Their major disadvantage is to be vulnerable to both *attribute value skew* (imbalance of the output of the first phase) and *join product skew* (imbalance of the output of local joins) [13, 11, 8]. The former affects immediate performance and the latter affects the efficiency of output or pipelined operations in the case of a multi-join.

The authors of [11] have identified the two best proposed solutions in conventional and sampling-based parallel join algorithms. Their study has allowed us to conclude [2] that all existing methods are sensitive to imbalance when applied multiple times because of JPS.
To address this problem, we introduced in [3] a deterministic data-redistribution algorithm with near-perfect balancing properties. It is adaptable to θ-join and efficient for multi-join. It dynamically computes *exact* frequency histograms to avoid *join product skew* (JPS). A scalable and portable cost analysis was made with the BSP model, leading to general predictions about the effect of relation histograms on performance. The analysis suggests a hybrid *frequency-adaptive* algorithm (*fa_join* algorithm) [3], dynamically combining histogram-based balancing with standard hashing methods. The *fa_join* algorithm avoids the slowdown usually caused by attribute value skew (AVS) and the imbalance of the size of local joins processed by the standard algorithms. It avoids AVS and JPS at the cost of extra processing. We analyzed this overhead both theoretically and experimentally and concluded that it does not penalize overall performance, even in the absence of skew.

The *fa_join* algorithm was thus designed to efficiently avoid the problem of AVS and JPS. However, its performance is less than ideal when computing the join of highly skewed relations because of unnecessary redistribution and communication costs. We introduce here a new parallel algorithm called *sfa_join* (Symmetric frequency adaptive join algorithm) to perform such joins. Its predictably low join-product and attribute-value skew make it suitable for repeated use in multi-join operations. Its performance is analyzed using the BSP cost model which predicts for it a near-linear speedup.

Bulk-Synchronous Parallelism (BSP) is a parallel programming model introduced by Valiant [14] to offer a high degree of abstraction like PRAM models and yet allow portable and predictable performance. The performance of a parallel architecture is characterized by 3 parameters expressed as multiples of the local processing speed: the number of processor-memory pairs p, the time l required for a global synchronization and the time g for collectively delivering a 1-relation (communication phase where every processor receives/sends at most one word). The network is assumed to deliver an h-relation in time $g * h$ for any arity h. The execution time of a parallel program is then the sum of 3 terms: W, the time spent by the most loaded processors on local computation between communication phases; $H = \sum_i g * h_i$ where h_i measures the i^{th} communication phase and $S * l$ where S is the number of such phases.

3 Data redistribution : A new approach

We first assume that relation R (resp. S) is partitioned among processors by horizontal fragmentation and the fragments R_i for $i = 1, .., p$ are almost of the same size on every processor, i.e. $|R_i| \simeq \frac{|R|}{p}$ where p is the number of processors. In the rest of this paper we use the following terminology for $T \in \{R, S\}$:

T_i denotes the fragment of relation T placed on processor i, $Hist(T)$ denotes the histogram of relation T with respect to the join attribute value, i.e. a list of pairs (v, n_v) where $n_v \neq 0$ is the number of tuples having the value v for the join attribute. The histogram is often much smaller and never larger than the relation it describes, $Hist(T_i)$ denotes the histogram of fragment T_i $Hist_i(T)$ is processor i's fragment of the histogram of T, $Hist(T)(d)$ is the frequency (n_d) of value d in relation T, $Hist(T_i)(d)$ is the frequency of value d in sub-relation T_i, and $|T|$ denotes the size of relation T.

We will outline the algorithm while giving an upper bound on the BSP execution time of each phase. A complete description can be found in [2]. Our redistribution algorithm is the basis for efficient and scalable processing. It proceeds in 5 phases:

Phase 1 : Creating local histograms

Local histograms $Hist(R_i)_{i=1,..,p}$ (resp. $Hist(S_i)_{i=1,..,p}$) of blocks R_i (resp. S_i) are created in parallel by a traversal of R_i (resp. S_i). In principle, this phase costs: $Time_{phase1} = O\left(\max_{i=1,..,p}\left(|R_i| + |S_i|\right)\right)$,
but in practice, the extra cost for this operation is negligible because the histograms can be computed on the fly while creating local hash tables.

Phase 2 : Creating the histogram of R ⋈ S

The first step is to create the histograms $Hist(R)$ and $Hist(S)$ by a parallel hashing of the histograms $Hist(R_i)$ and $Hist(S_i)$. The histograms $Hist(R_i)$ and $Hist(S_i)$ are first redistributed so that the complete histograms of R and S are evenly spread over the p processors. The cost of redistribution is : $Time_{phase2.a} = O(g * (|Hist(R)| + |Hist(S)|) + l)$. After hashing of destination addresses and communications are complete, each processor i merges the messages it received to constitute $Hist_i(R)$ (and $Hist_i(S)$) in time:
$Time_{phase2.b} = O(|Hist(R)| + |Hist(S)|)$.

While merging, processor i also retains a trace of the network layout of the values d in its $Hist_i(R)$ (resp. $Hist_i(S)$): this is nothing but the collection of messages it has just received. These data will be used in phase 4.
The fragment $Hist_i(R \bowtie S)$ is then computed on each processor i, by "intersecting" $Hist_i(R)$ and $Hist_i(S)$, in time:
$Time_{phase2.c} = O\big(\max_{i=1,..,p}(min(|Hist_i(R)|, |Hist_i(S)|))\big)$.
While creating the fragments of $Hist(R \bowtie S)$ we store, for each value $d \in Hist_i(R \bowtie S)$, an extra information called *index* ($index(d) \in \{0, 1, 2\}$). This information will permit us to decide if, for a given value d, the frequencies of tuples of relations R and S having the value d are greater (resp. lesser) than a threshold frequency f_0. It also permit us to choose dynamically the probe and build relation for each value d of the join attribute. This choice reduces to the minimum the global redistribution cost.
In the rest of this paper, we use the same threshold frequency as in the redistribution algorithm of *fa_join* [3], i.e. $f_0 = p * log(p)$. For a given value $d \in Hist(R \bowtie S)$,

- the value $index(d) = 0$, means that the frequency of tuples of relations R and S having the value d are less than the threshold frequency. i.e. $Hist(R)(d) < f_0$ and $Hist(S)(d) < f_0$,
- $index(d) = 1$ means that $Hist(R)(d) \geq f_0$ and $Hist(R)(d) \geq Hist(S)(d)$,
- $index(d) = 2$ means that $Hist(S)(d) \geq f_0$ and $Hist(S)(d) > Hist(R)(d)$.

At the end of this step 2.c, the histogram $Hist(R \bowtie S)$ is evenly partitioned between processors into fragments of size $|Hist_i(R \bowtie S)| \leq \min(\frac{|Hist(R)|}{p}, \frac{|Hist(S)|}{p})$ because every one is selected from a $\frac{1}{p}$-th of $Hist(R)$ and of $Hist(S)$ (obtained through hashing of the attribute values).

Every fragment $Hist_i(R \bowtie S)$ is then broadcasted to all processors in time: $Time_{phase2.d} = O\big(g * |Hist(R \bowtie S)| + l\big)$, and the histogram creation has therefore taken the sum of the above steps:
$Time_{phase2} = O\Big((1 + g) * (|Hist(R)| + |Hist(S)|) + l\Big)$.

Note that $Hist(R \bowtie S) \equiv Hist(R) \cap Hist(S)$ and $|Hist(R \bowtie S)|$ is generally very small compared to $|Hist(R)|$ and $|Hist(S)|$.

Phase 3 : Processing local semi-joins

To reduce the redistribution cost and thus the communication costs, we then compute the local semi-joins $\widetilde{R_i}$ and $\widetilde{S_i}$ as follows: $\widetilde{R_i} = R_i \ltimes S$ (resp. $\widetilde{S_i} = S_i \ltimes R$). This is done by a sequential scan of the fragment R_i (resp. S_i), on

each processor, by accessing the hash table of $Hist(R \bowtie S)$ in time:
$Time_{phase3} = O\Big(\max_{i=1,..,p}(|R_i| + |S_i|)\Big)$.

We observe that, unlike the hash-based algorithms where both relation R and S are redistributed, and as an improvement to the *fa_join* algorithm where only relations R and $S \ltimes R$ are redistributed, we will only redistribute $R \ltimes S$ and $S \ltimes R$ to perform the join operation $R \bowtie S$.

Phase 4 : Creation of communication templates

The attribute values which could lead to attribute value skew (those having high frequencies) are also those which may cause join product skew in standard algorithms. To eliminate this effect, we partition the histogram $Hist(R \bowtie S)$ into two sub-histograms: $\widehat{Hist}(R \bowtie S)$ and $\widetilde{Hist}(R \bowtie S)$ in the following manner: the values $d \in \widehat{Hist}(R \bowtie S)$ are associated to frequencies less than the threshold frequency. The tuples having these values for the join attribute, have no effect neither on AVS nor JPS. These tuples will be redistributed using a hash function. the values $d \in \widetilde{Hist}(R \bowtie S)$ are associated to frequencies greater than the threshold frequency f_0(i.e. $index(d) = 1$ or $index(d) = 2$). Tuples having these join attribute values, have an important effect on attribute value and join product skews. They will be redistributed using an appropriate redistribution algorithm to efficiently avoid both AVS and JPS.

To avoid data skew and thus balance the load among processors, redistribution of the semi-joins $\widetilde{R_i} = R_i \ltimes S$ (resp. $\widetilde{S_i} = S_i \ltimes R$) is necessary.

4.a On each processor i, the relation $\widetilde{R_i} = R_i \ltimes S$ (resp. $\widetilde{S_i} = S_i \ltimes R$) is divided into the three sub-relations $\widetilde{R_i'}$, $\widetilde{R_i''}$ and $\widetilde{R_i'''}$ (resp. $\widetilde{S_i'}$, $\widetilde{S_i''}$ and $\widetilde{S_i'''}$) in the following manner: $\widetilde{R_i} = \widetilde{R_i'} \cup \widetilde{R_i''} \cup \widetilde{R_i'''}$ (resp. $\widetilde{S_i} = \widetilde{S_i'} \cup \widetilde{S_i''} \cup \widetilde{S_i'''}$) where:

- All the tuples of relation $\widetilde{R_i'}$ (resp. $\widetilde{S_i'}$) are associated to values d such that $index(d) = 1$ (resp. $index(d) = 2$),
- All the tuples of relation $\widetilde{R_i''}$ (resp. $\widetilde{S_i''}$) are associated to values d such that $index(d) = 2$ (resp. $index(d) = 1$),
- All the tuples of relation $\widetilde{R_i'''}$ (resp. $\widetilde{S_i'''}$) are associated to values d such that $index(d) = 0$ (resp. $index(d) = 0$), i.e. the tuples which occur with low frequencies in both relations R and S.

This step costs at most : $Time_{phase4.a} = O(\max_{i=1,..,p}(|R_i| + |S_i|))$.

4.b Redistribution of $\widehat{R_i'}$ and $\widetilde{S_i'}$ is performed to avoid JPS because the values which could lead to attribute value skew (those having high frequencies) are those which often cause the join product skew. It avoids thus, the slowdown usually caused by attribute value skew and the imbalance of the size of local joins processed by the standard algorithms.

To this end, we first create a communication template: the list of messages which constitute the relations' redistribution. This step is performed jointly by all processors, **each one not necessarily computing the list of its own messages, so as to balance the overall process**. This step is completed (see [2]) in time: $Time_{phase4.b} = O(\max_{i=1,..,p} |\widetilde{Hist_i}(R \bowtie S)|)$. Phase 4, has

therefore taken the sum of the above two steps:
$Time_{phase4} = O\Big(\max_{i=1,..,p}(|R_i| + |S_i| + |\widetilde{Hist}_i(R \bowtie S)|)\Big)$.

Phase 5 : Data redistribution

5.a Redistribution of tuples having $d \in \widetilde{Hist}_i(R \bowtie S)$:

Every processor i holds, for every one of its local $d \in \widetilde{Hist}_i(R \bowtie S)$, the non-zero communication volumes it prescribes as a part of communication template. This step (see [2]) costs: $Time_{phase5.a} = O\left(g * |\widetilde{Hist}(R \bowtie S)| + |\widetilde{Hist}(R \bowtie S)| + l\right)$.

5.b Redistribution of tuples with values $d \in \widehat{Hist}_i(R \bowtie S)$:

To redistribute tuples having $d \in \widehat{Hist}_i(R \bowtie S)$, we first broadcast the fragments $\widehat{Hist}_i(R \bowtie S)$ to all processors in : $Time_{phase5.b} = O(g * |\widehat{Hist}(R \bowtie S)| + l)$, because every processor receives the whole $\widehat{Hist}(R \bowtie S)$ which is no more than it sends. At the end of this step, each processor, has local knowledge on tuples which will be redistributed using a hash function. At the end of steps 5.*a* and 5.*b*, each processor i, has local knowledge of how the semi joins $\widetilde{R}_i$ and $\widetilde{S}_i$ will be redistributed. Redistribution is then performed, in time: $O(g * (|\widetilde{R}_i| + |\widetilde{S}_i|) + l)$. Phase 5, has therefore taken:
$Time_{phase5} = O\Big(g * (|Hist(R \bowtie S)| + |\widetilde{R}_i| + |\widetilde{S}_i|) + |\widetilde{Hist}(R \bowtie S)| + l\Big)$,
and the complete redistribution algorithm costs $Time_{redist}$:

$$O\Big(\max_{i=1,..,p}(|R_i| + |S_i|) + g * (|\widetilde{R}_i| + |\widetilde{S}_i|) + (1+g) * (|Hist(R)| + |Hist(S)|) + l\Big).$$

We mention that, we only redistribute the semi-joins $\widetilde{R}_i$ and $\widetilde{S}_i$. Note that $|\widetilde{R}_i|$ (resp. $|\widetilde{S}_i|$) is generally very small compared to $|R_i|$ (resp. $|S_i|$) and $|Hist(R \bowtie S)|$ is generally very small compared to $|Hist(R)|$ and $|Hist(S)|$. Thus we reduce the communication cost to a minimum.

4 Symmetric frequency-adaptive join algorithm

To perform the join of two relations R and S, we first redistribute relations R and S using the above redistribution algorithm at the cost of $Time_{redist} =$

$$O\Big(\max_{i=1,..,p}(|R_i| + |S_i|) + g * (|\widetilde{R}_i| + |\widetilde{S}_i|) + (1+g) * (|Hist(R)| + |Hist(S)|) + l\Big).$$

Once the redistribution phase is completed, the semi-joins $\widetilde{R}_i$ (resp. $\widetilde{S}_i$) are partitioned into three disjoint relations as follow: $\widetilde{R}_i = \widetilde{R}'_i \cup \widetilde{R}''_i \cup \widetilde{R}'''_i$ (resp. $\widetilde{S}_i = \widetilde{S}'_i \cup \widetilde{S}''_i \cup \widetilde{S}'''_i$) as described in step 4.a of phase 4.
Taking advantage of the identity:

$$R \bowtie S = \Big(\bigcup_i \widetilde{R}'_i \bowtie \widetilde{S}''\Big) \cup \Big(\bigcup_i \widetilde{R}'' \bowtie \widetilde{S}'_i\Big) \cup \Big(\bigcup_i \widetilde{R}'''_i \bowtie \widetilde{S}'''_i\Big). \qquad (1)$$

Frequencies of tuples of relations $\widetilde{R}'_i$ (resp. $\widetilde{S}'_i$) are by definition greater than the corresponding (matching) tuples in relations $\widetilde{S}''_i$ (resp. $\widetilde{R}''_i$). Fragments $\widetilde{R}'_i$ (resp. $\widetilde{S}'_i$) will be thus chosen as *build* relations and $\widetilde{S}''_i$ (resp. $\widetilde{R}''_i$) as *probe* relations

to be duplicated on each processor. This improves over fa_join where all the semi-join $S \ltimes R$ is duplicated. It reduces communications costs significantly in asymmetric cases where both relations contain frequent & infrequent values.

To perform $R \bowtie S$, it is sufficient to compute the three following local joins $\widetilde{R'_i} \bowtie \widetilde{S''}$, $\widetilde{R''} \bowtie \widetilde{S'_i}$ and $\widetilde{R'''_i} \bowtie \widetilde{S'''_i}$ (cf. equation 1). To this end, we broadcast the fragments $\widetilde{R''_i}$ (resp. $\widetilde{S''_i}$) to all processors and complete local joins in all [2] in:

$$Time_{local-join} = O\Big(g * (|\widetilde{R''}| + |\widetilde{S''}|) + \max_{i:1,..,p}(|\widetilde{R'_i} \bowtie \widetilde{S''}| + |\widetilde{R''} \bowtie \widetilde{S'_i}| + |\widetilde{R'''_i} \bowtie \widetilde{S'''_i}|) + l\Big).$$

The global cost of the join of relations R and S using the symmetric frequency-adaptive join (*sfa_join*) algorithm is the sum of the redistribution cost with local join computation cost. It is of the order: $Time_{sfa-join} =$

$$O\Big(\max_{i=1,..,p}(|R_i| + |S_i|) + g * (|\widetilde{R_i}| + |\widetilde{S_i}|) + (1+g) * (|Hist(R)| + |Hist(S)|)$$
$$+ \max_{i:1,..,p}(|\widetilde{R'_i} \bowtie \widetilde{S''}| + |\widetilde{R''} \bowtie \widetilde{S'_i}| + |\widetilde{R'''_i} \bowtie \widetilde{S'''_i}|) + g * (|\widetilde{R''}| + |\widetilde{S''}|) + l\Big).$$

Remark : It can be shown that *sfa_join* algorithm has `optimal` asymptotic complexity when: $\max(|Hist(R)|, |Hist(S)|) \leq \max(\frac{|R|}{p}, \frac{|S|}{p}, \frac{|R \bowtie S|}{p})$,
This inequality often holds, because local joins have almost the same size and the histograms are often very small with respect to the relations.

5 Experimental results

The symmetric frequency-adaptive algorithm (sfa_join) for equi-join has been compared for speed-up and skew with standard hash-join (std_join) and the frequency-adaptive join algorithm (fa_join) introduced in [3]. Tests have used a Fujitsu AP-3000 shared-nothing architecture (based on Sparc Ultra-1 processors with 512Mb of memory) using MySQL-3.22 database servers for local relational processing and MPI-1.1 as message passing interface for communications. Frequencies of join attribute values have been taken to follow the Zipf distribution [16, 6] as it is the case in most database tests.

We have performed all tests on relations containing $100K$ tuples of 100 bytes each, attribute value skew has followed variations of the Zipf distribution's z parameter from 0 to 2 and the correlation factor has been varied from 0 to 100. The join selectivity[3] has been measured to vary from 0.00% to 0.15% as a result of variations of the join attribute correlation factor.

The speed-up test confirms the absence of a net overhead for sfa_join with respect to fa_join algorithm. It has been performed on relations with a slight attribute value skew (Zipf distribution's z parameter = 0.5) and of total join size $\simeq 3M$ tuples. The number of processors has varied from 1 to 12. Timings show an almost linear speed-up for both sfa_join and fa_join algorithms which are better than std_join algorithm (Fig. 1.a) *even for this low value of z*. Moreover the join product skew of std_join is relatively large with respect to relation size, while it is negligible for sfa_join and fa_join algorithms. This is shown by Fig. 1.b

[3] selectivity=$\frac{\|R \bowtie S\|}{\|R\| * \|S\|}$ where $\|T\|$ denotes the number of tuples of T.

where the JPS is measured by the maximum deviation of the local join result with respect to the average over all processors. As a result, when processing multiple join operations the cost of redistribution will be much smaller with sfa_join and fa_join algorithms than with the std_join algorithm. Figure 1.a shows that

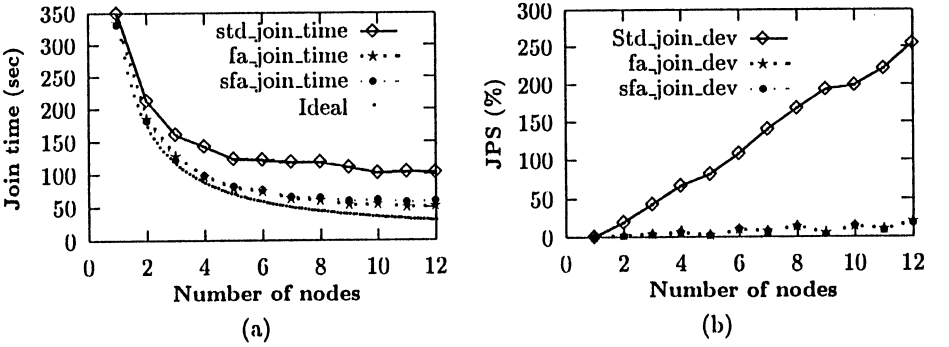


Fig. 1. Speed_up performance for small attribute skew.

sfa_join is slightly better than *fa_join*, this is mainly due to the redistribution cost which is lower for *sfa_join* compared to *fa_join*. We expect that the gap between the two algorithms will be large when computing very large relations. Currently, our tests are limited by the available disk space.

Two tests were performed to study the effect of attribute value skew on performance. The number of processors has been fixed to 12, relations R and S have been given sizes of $100K$ tuples. The first test used a fixed skew of 0.5 for R while varying skew in S from 0 to 2, yielding output relations of sizes between $700K$ and $15M$ tuples (i.e. selectivity between 0.007% and 0.15%). Timings show that sfa_join fa_join algorithms are vastly superior to the std_join algorithm while guaranteeing a balanced output. This test shows that sfa_join is slightly better than fa_join algorithm, which confirms its suitability in the presence of any amount of skew (Fig. 2).

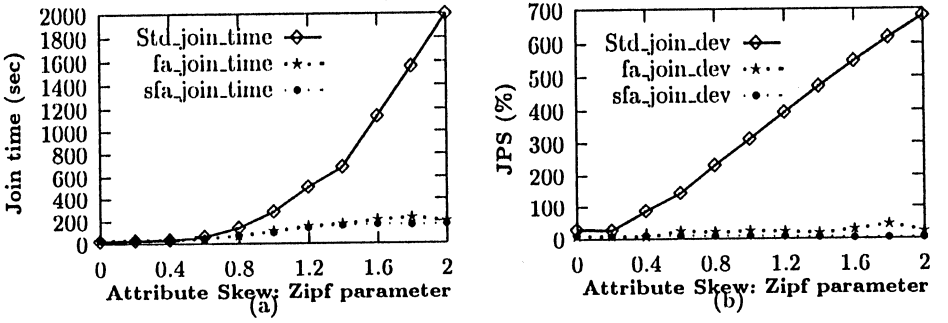


Fig. 2. Effect of attribute value skew for the probe relation.

The second test used a highly skewed relation R with a fixed skew of 2 while varying skew factor in S from 0 to 2, yielding output relations of sizes between $3.5M$ and $12M$ tuples. While varying skew factors using Zipf distributions, it is difficult to keep the join result approximately of constant size [4], so the join time in this test is proportional to join result size. Timings in Fig. 3 show that sfa_join algorithm always reaches linear speed-up whereas the timings for fa_join and std_join are proportional to JPS.
Figure 3.b shows that the sfa_join algorithm completely eliminates the problem of JPS. However the *fa_join* fails to avoid join product skew for a Zipf parameter close to 2. This can be explained by the fact that fa_join does not account for cases where a low frequency in the *build* relation combines with a high frequency in the *probe* relation. On the contrary, sfa_join handles the combination of this case and its inverse. The case of high frequencies in both relations is efficiently processed by both algorithms.

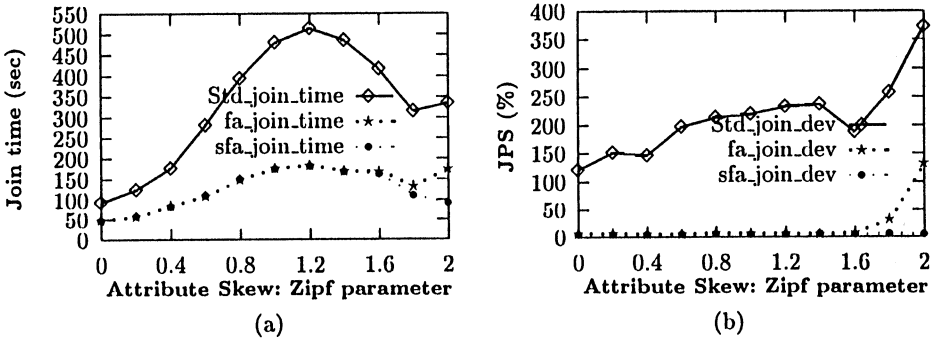


Fig. 3. Effect of high skew for both build and probe relations.

A last test measured the effect of join selectivity on performance in the same condition as the speed-up test (very small attribute value skew). The number of processors was fixed to 12, relations R, S were the same as in the speed-up test, the join selectivity has been made to vary from 0.005% to 0.07%, yielding output relations of sizes between $500K$ and $7M$ tuples. Results [2] show that sfa_join is faster than std_join independently of the join selectivity, while guaranteeing a very small *JPS*. This confirms and strengthens the speed-up test.

6 Conclusion

In this paper, we have introduced an efficient parallel join algorithm based on a "symmetric" sub-set replication allowing to reduce the communication costs while guaranteeing near perfect balancing properties.
The new algorithm is verified to improve on *fa_join* in two ways:1. The negative effect of non-correlated join attribute value distributions is completely eliminated at *no extra cost.* 2. The JPS, already very low for *fa_join*, is now completely eliminated which implies even faster multi-joins. The performance of this algorithm was analyzed using the BSP cost model which predicts a linear speedup.

The $O(\ldots)$ notation only hides small constant factors: they depend only on the implementation but neither on data nor on the BSP machine. The above tests confirm our theoretical predictions about processing overhead, attribute value and join product skews. They also show that sfa_join is never less efficient than fa_join and sometimes more, as predicted by theory.

References

1. M. Bamha and G. Hains. A frequency adaptive join algorithm for SN machines. *Journal of Parallel and Distributed Computing Practices*, 2000. To appear.
2. M. Bamha and G. Hains. A symmetric frequency-adaptive join algorithm for shared nothing machines. Research Report RR-LIFO-2000-03, LIFO, Université d'Orléans, 2000. ftp://ftp-lifo.univ-orleans.fr/pub/RR/RR2000/RR2000-03.ps.
3. M. Bamha and G. Hains. A self-balancing join algorithm for Shared Nothing machines. *In the Proc of the 10th International Conference on Parallel and Distributed Computing Systems, Las Vegas, Nevada*, October 1998.
4. David J. DeWitt, Jeffrey F. Naughton, Donovan A. Schneider, and S. Seshrdri. Practical Skew Handling in Parallel Joins. In *Proceedings of the 18th VLDB Conference, Vancouver, British Columbia, Canada*, 1992.
5. L. Harada and M. Kitsuregawa. Dynamic join product skew handling for hash-joins in shared-nothing database systems. In *Fourth International Conference on Database Systems for Advanced Applications*, pages 246–255, 1995.
6. Kian-Lee Tan Hongjun Lu. Dynamic and load-balanced task-oriented database query processing in parallel systems. In *Proceedings of the 3third Conf. Extending Data Base Technology,1992, pp.357-372*, 1992.
7. K. A. Hua and C. Lee. Handling data skew in multiprocessor database computers using partition tuning. In G. M. Lohman, A. Sernadas, and R. Camps, editors, *Proc. of the 17th International Conference on Very Large Data Bases*, pages 525–535, Barcelona, Catalonia, Spain, 1991. Morgan Kaufmann.
8. Hongjun Lu, Beng-Chin Ooi, and Kian-Lee Tan. *Query Processing in Parallel Relational Database Systems.* IEEE Computer Society Press, California, 1994.
9. H. Märtens. Skew-insensitive join processing in shared-disk database systems. *Proc. of Issues and Applications of Database Technology (IADT '98), Berlin*, 1998.
10. A. N. Mourad, R. J. T. Morris, A. Swami, and H. C. Young. Limits of parallelism in hash join algorithms. *Performance evaluation*, 20(1 / 3):301–316, May 1994.
11. Viswanath Poosala and Yannis E. Ioannidis. Estimation of query-result distribution and its application in parallel-join load balancing. *In: Proc. 22th Int. Conf. on Very Large Database Systems, VLDB'96, Bombay, India*, 1996.
12. Donovan A. Schneider and David J. DeWitt. A performance of four parallel join algorithms in a shared-nothing multiprocessor environment. *ACM SIGMOD*, 1989.
13. M. Seetha and Philip S. Yu. Effectiveness of Parallel Joins. *published in the IEEE, Trans. Knowledge and Data Enginneerings, Vol. 2, No 4, pp 410-424*, 1990.
14. Leslie Valiant. A Bridging Model for Parallel Computation,. *Communication of the ACM, Vol 33, No. 8.*, August 1990.
15. Annita N. Wilschut, Jan Flokstra, and Peter M.G. Apers. Parallel Evaluation of Multi-join Queries. *In the Proc. Of the ACM-SIGMOD, California*, 1995.
16. G. K. Zipf. Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology. *Reading, MA, Adisson-Wesley*, 1949.

An Interface to Databases for Flexible Query Answering: A Fuzzy-Set Approach

Grace Saulan Loo[1] and Kok-Huat Lee[2]

[1] Department of Management Science and Information Systems, The University of Auckland, Private Bag 92019 Auckland, New Zealand
g.loo@auckland.ac.nz

[2] Contact through Grace S Loo, g.loo@auckland.ac.nz

Abstract. This paper addresses a class of fuzzy queries with imprecise qualification. The imprecise qualification may contain a vague/imprecise expression, a linguistic term, or/and linguistic modifiers of the types (very) and (fairly). Single-attribute imprecise qualifications are studied in great length. A flexible α–cut technique is proposed for finding approximate answers for simple imprecise qualifications, and also for formulating an integrated structure to deal with multiple (very) and multiple (fairly) for answering composite qualifications. The issues on negation and complex queries are briefly discussed. The proposed α–cut approach is adopted in the design of a front-end objected-oriented intelligent interface to access a relational database via a general-purpose relational DBMS.

Keywords: α-cut, approximate, design, flexibility, fuzzy, imprecise, interface, linguistic, query modification.

1 Introduction

A conventional database system does not have the flexibility to handle *directly* fuzzy queries where the selection condition may contain a vague/imprecise term, or linguistic term or/and linguistic modifiers [5] and [6]. Semantic imprecision of vague or linguistic terms poses not a small problem in query processing. Earlier research activities used the concepts of fuzzy sets and the metric approach to resolve the problem of fuzzy queries; new processing techniques and operators were developed, and intelligent interfaces were designed to support approximate answering. The fuzzy-set approach is the main concern in this paper. The concept of fuzzy sets was introduced by [9] in 1965 to model imprecision. Much research has since then been conducted using this approach in various disciplines [1], [2], [4], [5], [6], [9], [10], and [11]. The α-cut notion of fuzzy set was employed in [5] as a flexible mechanism to search for approximate answers for some simple fuzzy queries. It was further used in [6] to develop a framework to process fuzzy queries where the selection condition contains linguistic modifier (very) or (fairly). This paper continues the work of [6] to present an integrated structure to deal with fuzzy queries using the α-cut concept, and to provide a simple flexible way of handling selection conditions with multiple modifiers of (very) and (fairly). The α-cut concept is incorporated into designing a user interface to access relational databases for flexible query answering. The main results are reported below.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 654–663, 2000.

2 Query Classification

Consider a class of fuzzy queries of the form: "Select all cases where X is q" where X is the single attribute under query, and q is a qualification requested to interrogate a domain of X. The selection condition 'X is q' is written here as X(q). Structurally q may be simple or composite. A simple qualification, re-denoted q_o for future reference, consists of a primary fuzzy/imprecise term which may involve a numeric quantity, a nominal term or a linguistic term, while a composite qualification is defined to contain a sequence of identical modifiers operating on q_o.

Six types of single-attribute qualifications are considered, namely, **S**imple **Nu**meric (SNu), **S**imple **No**minal (SNo), **S**imple **Li**nguistic (SLi), **C**omposite **Nu**meric (CoNu), **C**omposite **No**minal (CoNo), and **C**omposite **Li**nguistic (CoLi) qualifications.

(1) A SNu qualification q_o has the form: q_o = (mod y_o), where y_o is a crisp numeric target querying a numeric domain, and mod is a primary modifier operating on y_o to convert it into a fuzzy term q_o. Some mod_o operators suitable here are: 'close to', 'similar to', 'near to', 'about', 'approximate', 'approximately below', and 'approximately above'. For instance, the vague target (close to 75) in the condition SCORE(close to **75**) is a SNu qualification intended to query a numeric domain of the attribute SCORE.

(2) A SNo qualification q_o is defined as: q_o = (mod y_o), where the target y_o now refers to a nominal term, and mod is a primary modifier operating on y_o to convert it into a fuzzy term q_o. Some suitable mod operators here include: 'approximately', 'close to', 'similar to', and 'near to'. The target (close to orange) in COLOUR(close to orange), for instance, is a SNo qualification querying a nominal domain of the attribute COLOUR.

(3) A SLi qualification q_o has the form $q_o = lv_o$, where the target lv_o is a basic linguistic term requested to query a *numeric domain*. For instance, the linguistic term (old) in AGE(old) is a SLi qualification intended to query a numeric domain of the attribute AGE. Some additional examples of SLi qualifications are: (high), (low), and (heavy).

A *composite qualification*, re-denoted $q_{n(h)}$, is defined to contain n *identical* modifiers (h) operating in sequence on q_o; we write: $q_{n(h)} = [n(h); q_o]$. The n modifiers (h) are identical linguistic hedges of a given type such as 'very', 'fairly', 'rather', 'highly' or other fuzzy adverb. In this paper two groups of modifiers, denoted (very) and (fairly), are used. The group (very) represents a set of modifiers that have intensifying effect on the base q_o, while the group (fairly) refers to a set of modifiers that have relaxing effect on q_o. The qualification $q_{n(h)}$ is termed a CoNu, CoNo or CoLi qualification if its base q_o is a SNu, SNo or SLi qualification respectively. Some examples of selection condition with composite qualification are: (a) SCORE(*fairly* close to 80), (b) COLOUR(*very* red), (c) AGE(*very* young), and (d) SALARY(*very very* high).

3 Using α-Cut as an Acceptance Region of Approximate Answers

This paper presents an integrated framework based on the α-cut concept for finding acceptable approximate answers for single-attribute imprecise qualifications. Following [6] it is proposed to model an imprecise qualification q in X(q) as a fuzzy set. In this study a fuzzy set A in a universal set U is defined to have a membership function $\mu_A(x)$ such that $\mu_A(x) \in [0, 1]$ for every x in U. An element x with a high value of $\mu_A(x)$ has a high

'possibility' of being a member in A and vice versa. The support of A, denoted supp(A), is: supp(A) = $\{x \in U \mid \mu_A(x) > 0\}$. An α-cut of A, denoted A_α, is defined by:

$$A_\alpha = \{x \in U \mid \mu_A(x) \geq \alpha\} \text{ for } \alpha \in [0, 1] \quad (1)$$

The notion of α-cut is applied in this study as a flexible technique to search for acceptable approximate answers for imprecise qualifications. The measure α serves as 'index' of semantic proximity/similarity in the context of a given set of membership grades. A given α measures the *lowest level* of nearness relative to a given reference target that all acceptable cases must satisfy. A higher α indicates a higher level of nearness or similarity required for the acceptance of these cases for retrieval and vice versa. A higher α level yields a smaller α-cut, hence a narrower acceptance region. A lower α level with a wider acceptance region may be used if we are prepared to accept additional but 'less similar' cases which would otherwise be rejected when a higher α level is employed. Different α levels produce acceptance regions of different sizes for a given query. Therefore users can have the flexibility of interpreting a given imprecise term according to their own needs by merely choosing an appropriate α level to use.

One practical concern in using the α-cut technique is the need to determine appropriate membership function $\mu_A(x)$ for modelling a given imprecise term [3]. In general, membership grades may be assigned in a structured or unstructured manner. Their assignment, however, must adequately reflect the context of the problem under study. For a SNu or SNo qualification of the form (close to y_o) where y_o is a specific reference target, if the element-target distances $d(x, y_o)$ are known they can be converted into membership grades $\mu_A(x)$ using the following formula:

$$\mu_A(x) = [1 - \min (d(x, y_o)/D, 1)] \quad (2)$$

Here, $d_{min} < D \leq d_{max}$, with $d_{min} = \min\{d(x, y_o) | x \in U, y_o\}$, and $d_{max} = \max\{d(x, y_o) | x \in U, y_o\}$. The constant D is chosen subjectively to represent the maximum extent of imprecision that can be tolerated in a fuzzy set representation of (close to y_o). Any element x that has a distance $d(x, y_o)$ exceeding D has a zero $\mu_A(x)$ will be rejected as unacceptable for retrieval. Another conversion formula can be found in [8].

When the distances $d(x, y_o)$, hence the membership grades $\mu_A(x)$, are not known, an appropriate L-R-type fuzzy number/interval, if available, may be used as a membership function $\mu_A(x)$ for modelling a given imprecise qualification to interrogate a *numeric* domain. The choice of $\mu_A(x)$ is not unique. One or more competing choices may be available for representing the same imprecise qualification.

3.1 An α-cut of a L-R-type Fuzzy Interval (LRI)

L-R-types fuzzy numbers/intervals have been proposed in [7] for modelling SNu and SLi qualifications. This paper considers a L-R-type fuzzy interval A, denoted by: A = $(m_1, m_2, s, r; \delta)_{LRI}$, that has a membership function $\mu_A(x)$ given in Definition 1.

Definition 1. The LRI: A = $(m_1, m_2, s, r; \delta)_{LRI}$, with a shape parameter δ (> 0), has a membership function $\mu_A(x)$ defined by:

$$\mu_A(x) = \left[1+((x-m_1)/\delta)^2\right]^{-1}, \quad \text{if } m_1 - s \le x < m_1$$
$$= 1 \quad , \quad \text{if } m_1 \le x \quad m_2$$
$$= \left[1+((x-m_2)/\delta)^2\right]^{-1}, \quad \text{if } m_2 < x \quad m_2 + r$$
$$= 0 \quad , \quad \text{if } x < m_1 - s, \text{ or } x > m_2 + r$$

Here, supp(A) = $(m_1 - s, m_2 + r)$, with a core defined by the interval $[m_1, m_2]$. The left spread s (> δ) and right spread r (> δ) are equal. The left cross-over point is at $x = m_1 - \delta$, and the right cross-over point is at $x = m_2 + \delta$.

Theorem 1. An α-cut A_α of the LRI: $A = (m_1, m_2, s, r; \delta)_{LRI}$ is given by: $A_\alpha = [x_L, x_R]$,

$$\text{where: } x_L = m_1 - \delta\sqrt{\frac{1-\alpha}{\alpha}}, \text{ and } x_R = m_2 + \delta\sqrt{\frac{1-\alpha}{\alpha}}$$

The α-cut A_α given in Theorem 1 is obtained by setting $\mu_A(x)$ of Definition 1 equal to α and solving for x. See Figure 1. A_α depends only on δ and α, other than the quantities m_1 and m_2 requested by the query, and it is independent of the spreads s and r of $\mu_A(x)$.

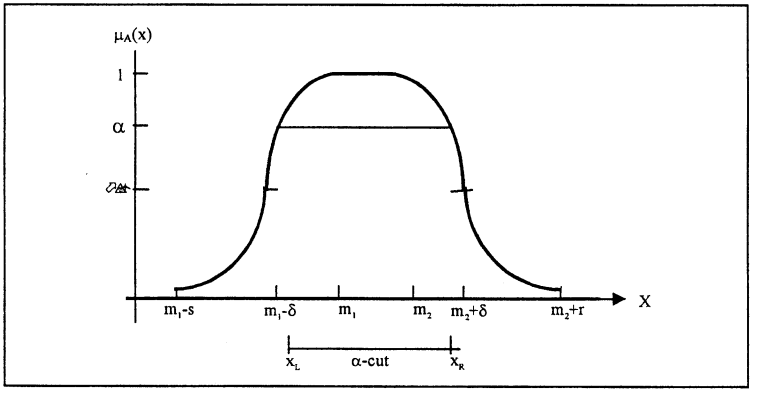


Fig. 1. An α-cut of $A = (m_1, m_2, s, r; \delta)_{LRI}$

Theorem 1 has its applications for a wide range of SNu and SLi qualifications. One application consists of modelling a SNu qualification: (approximately between y_1 and y_2) as $(y_1, y_2, s, r; \delta)_{LRI}$ by setting $m_1 = y_1$, and $m_2 = y_2$ in Theorem 1. Some other selective applications are reported in the following corollaries and results.

When the core interval $[m_1, m_2]$ contains only one point, say at x = m, the above LRI reduces to a L-R-type fuzzy number (LRN), re-denoted: $A = (m, s, r; \delta)_{LRN}$.

Corollary 1.1. A two-sided LRN: $A = (m, s, r; \delta)_{LRN}$ has an α-cut A_α given by:

$$A_\alpha = \left[m - \delta\sqrt{\frac{1-\alpha}{\alpha}} \ , \ m + \delta\sqrt{\frac{1-\alpha}{\alpha}} \right]$$

One application of Corollary 1.1 concerns with finding an α–cut for a two-sided SNu qualification of the form: (close to y_o) by setting $m = y_o$.

Corollary1.2. A left-sided LRN: $A = (m, s, r = 0; \delta)_{LRN}$ has an α-cut A_α provided by:

$$A_\alpha \geq m - \delta\sqrt{\frac{1-\alpha}{\alpha}}$$

A left-sided LRN is useful for representing a left-sided imprecise qualification and linguistic term such as (old) and (high). For instance, a left-sided SLi qualification, denoted (old), may be represented by: (old) = $(m, s, r = 0; \delta)_{LRN}$ to query a *numeric* domain of X. An α-cut of (old) can then be obtained from Corollary 1.2, where it is assumed that: $x \geq m$ belong to (old) *with certainty*. If X refers to human AGE, we may consider: $x \geq 60$ as "old" with certainty', and set $\delta = 5$, say, so that the crossover point of $\mu_{(old)}(x)$ occurs at $x = 55$. If $\alpha = 0.8$, say, then an 0.8-cut of (old) is: $(old)_{0.8} \geq 57.5$.

Corollary 1.3. A right-sided LRN: $A = (m, s = 0, r; \delta)_{LRN}$ has an α-cut A_α provided by:

$$A_\alpha \leq m + \delta\sqrt{\frac{1-\alpha}{\alpha}}$$

A right-sided LRN is suitable for modelling a right-sided imprecise qualification and linguistic term such as (young) and (low). For instance, a right-sided SLi qualification, denoted (low), may be modelled by: (low) = $(m, s = 0, r; \delta)_{LRN}$ to query a *numeric* domain of X. An α-cut of (low) can then be obtained from Corollary 1.3, where $x \leq m$ is viewed to belong to (low) *with certainty.* If X refers to the climatic temperature (in C°) of a region, we may consider: $x \leq 10$ as "low" with certainty, and set $\delta = 2$, say, so that: $x = 12$ is the crossover point of $\mu_{(low)}(x)$. Then an 0.2-cut of (low), say, is: $(low)_{0.2} \leq 14$.

Theorem 1 and its corollaries provide flexible procedures for finding acceptable approximate answers for SNu and SLi qualifications. These procedures are easy to implement *without the need to create unnecessary meta-database for query processing.*

4 Processing Composite Qualifications -- An α-Cut Solution

An integrated framework based on the notion of α–cut is proposed for processing composite qualifications of the form: $q_{n(h)} = [n(h); q_o]$; the modifier (h) may be an operator (very) or (fairly). A general power function of a base fuzzy set A is employed as a model to study the impact of each modifier (very) and (fairly) on A.

Definition. The p-th power of A, denoted A^p, is defined by: $A^p = (A)^p$, so that
for $x \in U$, $\mu_{A^p}(x) = [\mu_A(x)]^p$, where $p > 0$.

The index p in A^p is a positive number. Here, $supp(A^p) = supp(A)$.

Theorem 2. $(A^p)_\alpha = (A)_\beta$, where $\beta = \alpha^{1/p}$ for $p > 0$, and $0 < \alpha$ 1.

Theorem 2 is due to [6]. It states that an α-cut of A^p is equivalent to the β-cut of A, where $\beta = \alpha^{1/p}$. It is true for any base A, and any positive number p. It provides a simple procedure for deriving an α-cut of A^p from that of A, without the need to determine the actual form of A^p. It is used in this paper to develop an integrated structure for handling composite qualifications of the form: $q_{n(h)} = [n(h); q_o]$.

Theorem 3. An α–cut of $q_{1(h)} = (q_o)^p$, with $p > 0$, is given by:

$$(q_{1(h)})_\alpha = (q_o)_\beta, \text{ where } \beta = \alpha^{1/p}.$$

Theorem 3 states that an α-cut of $q_{1(h)}$ is equivalent to a β-cut of q_o, where $\beta = \alpha^{1/p}$.

4.1 Using an EME Model for Establishing α-Cut

To construct an α-cut for $q_{n(h)} = [n(h); q_o]$, an **E**qual-**M**odification-**E**ffect (EME) Model, is proposed here on the assumption that the n modifiers (h) exert *equal* modification effects at each stage of query modification so that: $q_{j(h)} = [q_{(j-1)(h)}]^t$, for $t > 0$, and all (j = 1, 2, ... , n), with $q_{0(h)} = q_o$. Evidently, $q_{n(h)} = (q_o)^{t^n}$. Applying Theorem 3 produces an α-cut for $q_{n(h)}$ as stated in Theorem 4 below.

Theorem 4. For $q_{n(h)} = [n(h); q_o]$, if the recursive relation: $q_{j(h)} = [q_{(j-1)(h)}]^t$ is defined for $t > 0$ and all j = 1, 2, ..., n, with $q_{0(h)} = q_o$, then:

$$q_{n(h)} = (q_o)^{t^n}, \text{ and } (q_{n(h)})_\alpha = (q_o)_\beta, \text{ where } \beta = \alpha^{t^{-n}}.$$

The modifiers (h) may be (very) or (fairly). The parameter t in the relation: $q_{j(h)} = [q_{(j-1)(h)}]^t$ is chosen such that: $t > 1$ for (h) = (very), and $0 < t < 1$ for (h) = (fairly); the value of t = 2 may be used for the CON operator and t = 0.5 for the DIL operator of [10]. The EME model provides an integrated structure to answer composite qualifications $q_{n(h)}$, regardless of the fuzzy set representation of the base q_o. The membership grades of q_o may be assigned in an unstructured manner, or be modeled as a L-R-type fuzzy number/interval.

4.2 Answering CoNu and CoNo Qualifications

Corollary 4.1. If $[n(very); (close\ to\ y_o)] = (close\ to\ y_o)^{t^n}$, where $t > 1$, then

$$[n(very); (close\ to\ y_o)]_\alpha = (close\ to\ y_o)_\lambda, \text{ where } \lambda = \alpha^{t^{-n}}.$$

Moreover, in Corollary 4.1, if: $(close\ to\ y_o) = (m = y_o, s, r; \delta)_{LRN}$, then

$$[n(very); (close\ to\ y_o)]_\alpha = \left[y_o - \delta\sqrt{\frac{1-\lambda}{\lambda}},\ y_o + \delta\sqrt{\frac{1-\lambda}{\lambda}} \right], \text{ where } \lambda = \alpha^{t^{-n}}$$

Corollary 4.2. If $[n(fairly); (close\ to\ y_o)] = (close\ to\ y_o)^{t^n}$, for $0 < t < 1$, then

$$[n(fairly); (close\ to\ y_o)]_\alpha = (close\ to\ y_o)_\gamma, \text{ where } \gamma = \alpha^{t^{-n}}$$

4.3 Answering CoLi Qualifications

Corollary 4.3. Let (old) = (m, s, r = 0; $\delta)_{LRN}$ over a domain interval of X, where $x \geq m$ are treated as "old" with certainty. If (very old) = $(old)^t$, for $t > 1$, then

$$(very\ old)_\alpha = (old)_\beta \text{ , and } (very\ old)_\alpha \geq m - \delta\sqrt{\frac{1-\beta}{\beta}} \text{ , where } \beta = \alpha^{1/t}.$$

Corollary 4.3 is a direct result of Theorem 3 and Corollary 1.2.

Corollary 4.4. Let (low) = (m, s = 0, r; $\delta)_{LRN}$ over a domain interval of X, where $x \leq m$ are treated as (low) with certainty. If (fairly low) = $(low)^t$, for $0 < t < 1$, then: $(fairly\ low)_\alpha = (low)_\beta$, giving $(low)_\beta \leq m + \delta\sqrt{\frac{1-\beta}{\beta}}$, for $\beta = \alpha^{1/t}$.

Corollary 4.4 is the direct result of Theorem 3 and Corollary 1.3.

5 Processing Negation of a Qualification

The negation of a qualification q has the effect of altering the meaning of q in an opposite sense. The negation of q, denoted (not q) or ($\neg$ q), is imprecise if q is imprecise where q may be a SNu, SNo, SLi, CoNu, CoNo, or CoLi qualification. Some examples of negation are: ($\neg$ close to 50), ($\neg$ red), ($\neg$ old), ($\neg$ very low), and ($\neg$ very very heavy). Let $\mu_{(q)}(x)$ be a membership function of the fuzzy set (q) that represents the qualification q. A membership function $\mu_{(\neg q)}(x)$ of ($\neg$ q) is then defined as:

$$\mu_{(\neg q)}(x) = 1 - \mu_{(q)}(x), \quad \text{for } x \in U \tag{3}$$

An α-cut of ($\neg$ q), denoted $(\neg q)_\alpha$, may be computed directly from an explicit form of the negation ($\neg$ q), if available, or determined as the complement to an $(1 - \alpha)$-cut of (q), i.e.,

$$(\neg q)_\alpha = U - (q)_{(1-\alpha)}, \text{ for } x \in U \tag{4}$$

The fuzzy set (q) and its negation ($\neg$ q) are not mutually exclusive in the universe set U. The α-cut of the negation ($\neg$ q) and the $(1-\alpha)$-cut of (q) are however mutually exclusive and collectively exhaustive in U. This result provides a simple way of computing the needed α-cut of ($\neg$ q) without using any explicit form of its fuzzy set representation. It is applicable when q is a SNu, SNo, SLi, CoNu, CoNo or CoLi qualification, or a complex target that involves more than one single-attribute selection condition.

6 Answering Complex Queries

A complex query with a multi-attribute selection condition Q_n has a form:

$$Q_n = [X_1(q_1)\ ®_1\ X_2(q_2) \ldots ®_{n-1}\ X_n(q_n)] \tag{5}$$

Q_n has n simple selection conditions: $X_t(q_t)$, (t = 1, 2, ... , n). Each target q_t is defined over the domain of one attribute X_t, and it may be a SNu, SNo, SLi, CoNu, CoNo, or CoLi qualification. Each of the (n-1) identical connectives $®_t$ is either entirely a Boolean $\cap$ or $\cup$ operator. An overall acceptance region $AR[Q_n]$ of Q_n is given by:

$$AR[Q_n] = [AR(q_1) ®_1 AR(q_2) ... ®_{n-1} AR(q_n)] \quad (6)$$

where $AR(q_t)$ is an acceptance region for each individual $X_t(q_t)$. A solution to Q_n may be obtained by applying one of the following two procedures 1 and 2.

Procedure 1. Model each of the n simple selection condition $X_t(q_t)$ by a fuzzy set A_t, and obtain an α_t-cut $(A_t)_{\alpha_t}$ of A_t as an acceptance region for each individual $X_t(q_t)$; that is, $AR(q_t) = (A_t)_{\alpha_t}$. Invoking Eq. (6) produces $AR[Q_n]$ as a solution to Q_n, where

$$AR[Q_n] = [(A_1)_{\alpha_1} ®_1 (A_2)_{\alpha_2} ... ®_{n-1} (A_n)_{\alpha_n}] \quad (7)$$

The solution $AR[Q_n]$ of Eq. (7) is the direct result of replacing the acceptance region $AR(q_t)$ with the corresponding α_t-cut $(A_t)_{\alpha_t}$ in Eq. (5). With this solution, different α_t levels may be specified for individual α_t-cuts $(A_t)_{\alpha_t}$, and the answer space for Q_n can be altered by changing the α_t level of one or more of the α_t-cuts $(A_t)_{\alpha_t}$. This feature provides a flexibility in using different α_t levels for different component α_t-cuts $(A_t)_{\alpha_t}$ with the preferred option of using a lower α_t level for a less stringent selection condition.

Procedure 2. Model each $X_t(q_t)$ by a fuzzy set A_t, and define an overall fuzzy set to represent the complex target Q_n as a combination of these component fuzzy sets A_t within the structure specified in Eq. (5), by applying standard fuzzy set operations such as those suggested in [9]. An α-cut of the resultant fuzzy set (Q_n) is then constructed as an acceptance region for the complex target Q_n.

When the complex target Q_n is *not fully* satisfied, empty answers will result. The problem of empty answers may however be resolved by modifying the query in one of the following three ways: (1) Apply a removal process. (2) Employ a relaxation process. (3) Use a combination of the removal and relaxation processes [7]. In the relaxation process, two issues emerge: (i) The search scope of approximate answers needs to be enlarged. (ii) The extent by which a particular selection condition $X_t(q_t)$ can be relaxed need to be specified. With the proposed α-cut technique, however, issue (i) of expanding the search scope for approximate answers can be resolved by using a lower α level. Issue (ii) can be resolved by widening the support of the fuzzy set that represents the selection condition through altering the value of the parameters such as δ that characterize the L-R-type fuzzy number/interval used [7].

7 A Flexible Query Answering Interface to Relational Databases

The necessity to have flexible query answering interfaces to databases has been recognised by researchers and DBMS vendors for the past two decades. Several interface systems have been produced. These systems often require pre-precompiled metadata

with unduly increase in the size of the database and the general processing overheads. Moreover, they are somewhat inflexible. Users cannot *easily* change their criteria or employ their own definitions on the size of the acceptance region according to their information needs. This section presents a proposed front-end intelligent information interface operating through a relational DBMS to access a relational database with an aim to enhance/enrich the query answering capability of the database to deal with the fuzzy queries of the types discussed in this study. The proposed system, *ISKREOT*, is designed to incorporate an expert system with object technologies; the acronym *ISKREOT* stands for "*I*ntelligent *S*ystem for *K*nowledge *R*epresentation using *E*xpert system and *O*bject *T*echnology". *ISKREOT* uses *GoldWorks* to provide the required expert system, and *ORACLE* to provide the desired relational DBMS. The object-oriented expert shell of *ISKREOT* uses the expert database approach of external enhancement with tight coupling to reconcile the relational and object-oriented models. The design of *ISKREOT* uses the re-engineering approach to enable the host database to behave as an *intelligent object-oriented database system,* providing application-independent intelligent capability with user-friendly services for handling fuzzy queries. An overview of query processing design of *ISKREOT* is given in Figure 2.

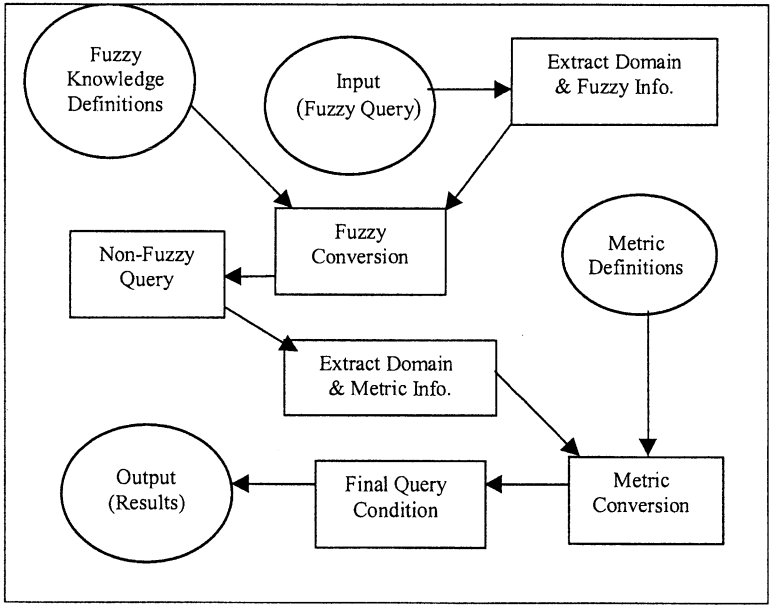


Fig. 2. An overview of query processing design of *ISKROET*

In the *ISKROET* environment, when a user's fuzzy query is received two processes are activated. The fuzzy query first goes through the system's *fuzzy conversion* process utilizing the result of domain extraction and fuzzy information, and the predefined fuzzy knowledge definitions to produce a corresponding *non-fuzzy query*. The resultant non-fuzzy query then goes through the system's metric conversion process utilizing the result

of domain extraction and metric information, and the available metric definitions to produce a final query condition which is then evaluated for finding approximate answers for the original fuzzy query. The prototype *ISKROET* is user-friendly. An *ISKREOT* user can interactively change the size of α, hence varying the search scope of acceptance region, and modify the definitions of fuzzy qualifiers (simple or complex) to suit the user's own information needs. The changes and execution of the fuzzy query are done *on-the-fly*, without the need to create and use a large meta-database.

Many fuzzy queries and some standard SQL formatted queries were tested on a sample relational database. The test run results are re-assuring with positive evidence in support of the validity and ease of the proposed α–cut technique for implementing flexible query answering; see [7]. In passing, the popular *Internet* has a heterogeneous mix of general and professional users who often seek to access large files on line. The proposed α–cut technique will provide a useful way of handling fuzzy queries *on-the-fly* with the *flexibility* to interpret linguistic terms/expressions in accordance with the information needs of different application users.

References

1. Andreasen, T. and O. Pivert, 1994. 'On the weakening of fuzzy relational queries'. In Z. W. Ras and M. Zemankova (eds.), *LNAI 869: Methodologies for Intell. Sys., Proc. 8th Int. Symp., ISMIS'94, Charlotte, North Carolina, October 1994.* Springer-Verlag, pp. 144-153.
2. Bezdek, J. C. and S. K. Pal (eds.), 1992. *Fuzzy Models for Pattern Recognition.* - Methods That Search for Structures in Data. IEEE Press, New York.
3. Bobrowicz, O., C. Choulet, A. Haurat, F. Sandoz and M. Tebaa, 1990. 'A method to build membership functions - Application to numerical/symbolic interface building'. In B. Bouchon-Meunier, R. R. Yager and L. A. Zadeh (eds), 1991. *LNCS 521: Uncertainty in Knowledge Bases, Proc. 3rd Int. Conf., on Info. Processing and Management of Uncertainty in Knowledge-Based Sys., IPMU '90, Paris, France, July 1990.* Springer-Verlag, pp. 136-142.
4. Chu, W. W., and Q. Chen, 1992. 'Neighbourhood and associative query answering'. *J. Intell. Info. Sys.*, V 1. Kluwer Academic Publishers, Boston, 1992, pp. 355-382.
5. Loo, S. L., T. S. Dillon, J. Zeleznikow and K. H. Lee, 1994. 'Two approaches for answering inadequate queries - Empirical project IDB-KROOM'. In H. Larsen and T. Andreasen (eds.), *Proc. Workshop on Flexible Query-Answering Systems', FQAS94, Roskilde, Denmark, November 1994.* Roskilde University Press, pp. 127-146.
6. Loo, S. L., T. S. Dillon, J. Zeleznikow and K. H. Lee, 1996. 'Enhancing query processing of information systems'. In Z. W. Ras and M. Michalewicz (eds.), *LNAI 1079: Foundations of Intell. Sys., Proc.9th Int. Symp., ISMIS'96, Zakopane, Poland, June 1996.* Springer, pp. 386-397.
7. Loo, S. L, 1998. *Intelligent Databases.* Unpublished report. Department of MSIS, University of Auckland.
8. Prade, H. and C. Testemale, 1989. 'The Possibilistic Approach to the Handling of Imprecision in Database Systems'. In *Bull. IEEE Computer Society, Data Engineering*, V 4.10, June 1989, pp. 4-10.
9. Zadeh, L. A., 1965. 'Fuzzy sets'. In *Info. and Control*, V 8.3, June 1965, pp. 338-353.
10. Zadeh, L. A., 1973. 'Outline of a new approach to the analysis of complex systems and decision processes'. In *IEEE Trans. on Sys., Man And Cybernectics*, SMC-3, pp. 28-44.
11. Zadeh, L. A., 1978. 'Fuzzy sets as a basis for the theory of possibility'. *Fuzzy Sets and Sys.*, V 1, 1978, pp. 3-28.

Lattice-structured Domains, Imperfect Data and Inductive Queries

Sally Rice[1] and John F. Roddick[2]

[1] School of Computer and Information Science, The Levels Campus,
University of South Australia, Mawson Lakes 5095, South Australia.
rice@cis.unisa.edu.au
[2] School of Informatics and Engineering, Flinders University of South Australia,
GPO Box 2100, Adelaide 5001, South Australia.
roddick@cs.flinders.edu.au

Abstract. The relational model, as proposed by Codd, contained the concept of relations as tables composed of tuples of single valued attributes taken from a domain. In most of the early literature this domain was assumed to consist of elementary items such as simple (atomic) values, defined complex data types or arbitrary length binary objects. Subsequent to that the nested relational or non-first normal form model allowing set-valued or relation-valued attributes was proposed. Within this model an attribute could take multiple values or complete relations as values. This paper presents a further extension to the relational model which allows domains to be defined as a hierarchy (specifically a lattice) of concepts, shows how different types of imperfect knowledge can be represented in attributes defined over such domains, and demonstrates how lattices allow the accommodation of some forms of inductive queries. While our model is applied to flat relations, many of the results given are applicable also to nested relations. Necessary extensions to the relational algebra and SQL, a justification for the extension in terms of application areas and future research areas are also discussed.

1 Introduction

Attribute values in relational databases which are not numeric, spatial or temporal are commonly interpreted as labels with no semantic connection between them beyond that of simple ordering. For example, consider the attribute domain {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday, weekday, weekend}. As well as the order of the days of the week, there is also an inclusion relationship between Sunday and weekend (since Sunday is part of the weekend). This type of relationship can be expressed in a concept lattice, which can be represented in the database as a domain concept relation – a simple child-parent binary relation between pairs of labels from the domain [10, 11]. Kedad and Métais [4] have looked at hierarchical domains, but their approach is limited to trees.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 664–674, 2000.

Mannila [5] defines inductive databases as databases that contain inductive generalisations about data. The inductive queries discussed in this paper, although relevant to these inductive generalisations (for example, see [9–11]), are not restricted to inductive databases, but can be used in any database which has the ability to represent hierarchical domains and store imperfect data.

There are many terms used in the literature for different types of imperfect information. Parsons [6] offers a classification of imperfect information into five categories: uncertainty (lack of information), imprecision (lack of granularity), vagueness (the fuzziness implied by terms such as 'young' or 'short'), incompleteness (lack of relevant information) and inconsistency (contradictory information). We wish to consider three cases of imperfection which we present in detail in the next section of this paper. The type of imperfect information captured by the use of weekend instead of Sunday we term *incomplete*. Our first example using a lattice built on geographic locations gives rise to imperfect information of this type. Our second example uses a lattice of temporal intervals, with intervals at a higher level of the lattice containing those at lower levels. This type of information we term *imprecise*. Our third example is of *inconsistent* information, where one piece of data contradicts another. The term uncertainty in this paper is reserved for describing the effect of matching domain values at different levels of the hierarchy with the attribute value in an inductive query. Parson's uncertainty and vagueness are not addressed explicitly, but vagueness does not differ significantly from the incomplete information example, and uncertainty could be modelled using inconsistency if information with 70% certainty was interpreted as weights of 0.7 on the information and 0.3 on its negation.

This paper explores the use of lattices for describing the three examples, then introduces *inductive queries* (queries on attributes defined on lattice-structured domains) and discusses the uncertainty inherent in these queries. Necessary functions that operate on lattices, such as H and L [9] which retrieve all the concepts in the lattice above and below a given concept, are defined and an algorithm which uses these functions to divide the elements in the lattice into three sets corresponding to the truth values t (true), f (false) and u (uncertain) for a given inductive query is given. The implications of inductive queries for relational algebra operations is investigated, and some extensions to the algebra are proposed. These extensions, and other changes to the definition and data update facilities of query languages for inductive queries are discussed within the framework of an implemented query language (SQL).

The paper closes with a discussion of application areas and future research for this work.

2 Lattice-structured Domains

The structure of any hierarchical domain can be described by a domain concept relation DS = (element, concept), where each tuple describes a relation between two elements of the domain – the first element belongs to the concept that the second element describes. Consider a domain D with the twelve ele-

ments {Americas, North America, Central America, South America, USA, Canada, Mexico, Nicaragua, Brazil, Chile, northern hemisphere, southern hemisphere}. Figure 1 (a) shows a tree-structured domain built from ten of them, describing the geographical relationship between the elements. Each element at the lower levels of the structure has a single parent at the previous level, leading to unique paths from any node to the root of the structure to which it belongs.

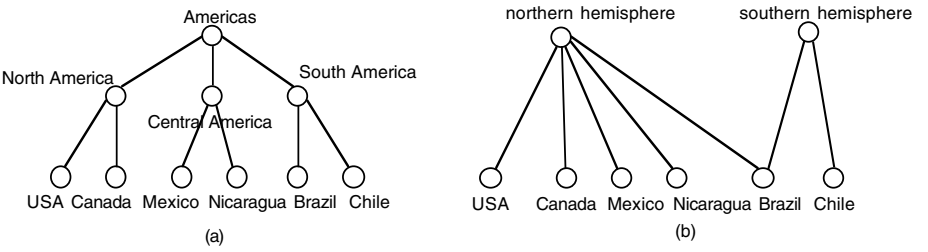


Fig. 1. (**a**) Tree-structured domain and (**b**) lattice-structured domain

Allowing children to have multiple parents in a structure changes the structure from a tree to a lattice: a simple example is that shown in Fig. 1 (b), where one child, Brazil, has two parents. Table 1 shows a domain concept relation combining the two hierarchical structures shown in Fig. 1. Top level concepts (Americas, northern hemisphere and southern hemisphere) are identified by never appearing in the element column, base level concepts (the names of the individual countries) by never appearing in the concept column. Middle level concepts appear in both columns.

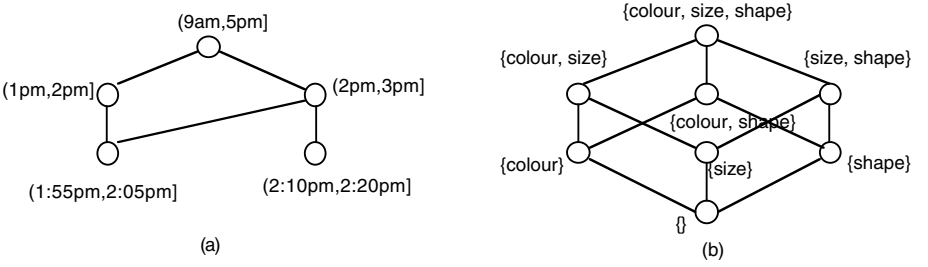


Fig. 2. (**a**)Lattice of intervals and (**b**) lattice of attribute sets

This example shows that information at different levels of granularity can be combined into a single lattice-structured domain, allowing the use of incomplete (i.e. coarser grained) information in relations defined on this domain. Lattices can also be used for imprecise information represented as intervals, by labelling

Table 1. Domain concept relation

element	concept
USA	North America
Canada	North America
Mexico	Central America
Nicaragua	Central America
Brazil	South America
Chile	South America
North America	Americas
Central America	Americas
South America	Americas
USA	northern hemisphere
Canada	northern hemisphere
Mexico	northern hemisphere
Nicaragua	northern hemisphere
Brazil	northern hemisphere
Brazil	southern hemisphere
Chile	southern hemisphere

each node in the lattice with an interval as shown in Fig. 2 (a). The idea of representing interval data in lattices is not new, see [2] for example. Similarly, a lattice whose nodes are sets of attributes can be defined for any relation and used to identify inconsistent information in tuples of the relation. If the same object is described as both red, small and square, and blue, small and square, we can represent this inconsistency by the node marked {colour} in Fig. 2 (b).

3 Uncertainty in Inductive Queries

For the domain shown in Fig. 1 (b), the semantic interpretation appears to be that Canada, USA, Nicaragua and Mexico are wholly within the northern hemisphere, Chile is wholly within the southern hemisphere, and Brazil is split between both hemispheres. However, the way the domain is used may change the meaning, so that uncertainty is inherent in relations using this domain. Consider relation R = (name, location) shown in Table 2, where *location* is defined on the lattice-structured domain described in Table 1. If the locations refer to a specific, if unknown, address, then this relation says that John lives somewhere in Brazil, and almost certainly either in the northern hemisphere or the southern hemisphere, but not both. This introduces a degree of uncertainty into whether or not conditions on these elements hold or not. Given any condition C on attribute

Table 2. Relations R, S and Q using a lattice-structured domain

Relation R		Relation S		Relation Q	
name	location	name	location	location	language
John	Brazil	Susan	South America	Canada	English
Susan	Chile	Peter	Brazil	Canada	French
Peter	southern hemisphere			USA	English
Betty	Nicaragua			Mexico	Spanish
				Nicaragua	Spanish
				Brazil	Portugese
				Chile	Spanish

A defined on lattice-structured domain D, that condition can be categorised as *known to be true*, *uncertain* (may or may not be true), or *known to be false*. For example, the condition `location = 'southern hemisphere'` is known true for Susan, known false for Betty, and uncertain for John. These categories create three mutually exclusive sets T, U and F for the elements $e \in D$, where $T = \{e|C \text{ true}\}$, $U = \{e|C \text{ uncertain}\}$ and $F = \{e|C \text{ false}\}$. In addition to these sets, we can define another three (useful) groupings by combining them in pairs. This leads to the definition of the six inductive conditions shown in Table 3. The symbol ? indicates that condition C may be true (i.e. true or uncertain), and the symbol ! that the value of C is not uncertain (i.e. true or false). The

Table 3. Inductive conditions

condition	meaning	included elements
C	C true	T
$\neg C$	C false	F
$\neg(!C)$	C uncertain	U
$?C$	C true or uncertain	$T \cup U$
$\neg(?C)$	C uncertain or false	$U \cup F$
$!C$	C true or false	$T \cup F$

inductive conditions degenerate as expected where no uncertainty exists. In the case of unstructured domains $U = \phi$, so $?C$ and $\neg(?C)$ are equivalent to C and $\neg C$ respectively, and $!C$ is always true and $\neg(!C)$ always false.

As well as the uncertainty introduced in relation R by saying that John lives somewhere in Brazil, there is another type of uncertainty in saying that Peter

lives in the southern hemisphere. This means that Peter might live in either Chile or Brazil, but not both. One type of uncertainty is introduced because of the use of an element in the tuple at a higher level than that in the query (type 1 uncertainty), and another because of the use of an element in the query that is a member of more than one higher level concept (type 2 uncertainty – this does not arise if structures are limited to trees).

Consider the conditions C_1 (`location = 'Brazil'`) and C_2 (`location = 'northern hemisphere'`), where C_1 leads to type 1 uncertainty, and C_2 to type 2. Table 4 shows the names in sets T, U and F for each condition. Only T

Table 4. Sets T, U and F for conditions C_1 and C_2

	C_1	C_2
T	{John}	{Betty}
U	{Peter}	{John}
F	{Susan, Betty}	{Susan, Peter}

and U need to be calculated by examining the lattice defined by relation DS, since $F = \Omega - T - U$. Once T and U have been determined, the type of uncertainty is no longer of importance. Consider a simple condition of the type $a = e$, where a is an attribute defined on a structured domain, and e is an element from that domain. For domains with type 2 uncertainty, $T = \{t | t \in R \wedge (a = e \vee a$ must be a descendent of $e\}$ and $U = \{t | t \in R \wedge (a$ is an ancestor of $e \vee a$ may be a descendent of $e)\}$. For domains with only type 1 uncertainty, if there are no lower level concepts with multiple parents, the algorithm needed to construct sets T and U can be simplified. Structurally, the difference between type 1 and type 2 uncertainty for domain value e is whether any descendents of the node e in the lattice have a path to the top of the lattice which does not go through e.

In the case of compound conditions such as $C_a \wedge C_b$ or $C_a \vee C_b$, the set to which a given tuple belongs can be determined by considering the 3-valued logic truth tables shown in Table 5. Here t, u and f represent the truth values true, uncertain and false respectively.

4 Determining Truth Values for Inductive Queries

In order to construct sets T and U, we need to be able to identify higher and lower level concepts for elements from a structured domain. Let e be an elemental value from a lattice-structured domain D whose structure is defined by domain concept relation DS. $H(e)$ is the set of higher-level concepts of e, i.e. $H(e) = \{c | c \in D \wedge c$ is an ancestor of $e\}$, and $L(e)$ is the set of lower-level concepts of

Table 5. 3-valued logic truth tables

C	$\neg C$
t	f
u	u
f	t

$C_a \vee C_b$	t	u	f
t	t	t	t
u	t	u	u
f	t	u	f

$C_a \wedge C_b$	t	u	f
t	t	u	f
u	u	u	f
f	f	f	f

e, i.e. $L(e) = \{c | c \in D \wedge c$ is a descendent of $e\}$. If e is a base concept, then $L(e) = \phi$, and if e is a top-level concept, then $H(e) = \phi$. If D is an unstructured domain, then $L(e) = H(e) = \phi$ for all e. For our example, H(South America) = {Americas}, L(South America) = {Brazil, Chile}, H(Brazil) = {South America, Americas, southern hemisphere, northern hemisphere} and L(Brazil) = ϕ.

Another useful hierarchical function $A(e_1, e_2)$ is defined as the set of all elements from D which are ancestors of e_2 which are not related to e_1. If $\{e_2 | e_2 \in L(e_1) \wedge A(e_1, e_2) \neq \phi\}$ is not empty, then type 2 uncertainty exists for a query using e_1. The function ancestor is used to formally define the set-valued functions H, L and A, and thus determine the sets T and U for a given domain element value e. These functions are defined below.

$$
\begin{aligned}
\text{ancestor}(x, y) &= (x, y) \in DS \vee ((x, z) \in DS \wedge \text{ancestor}(z, y)) \\
H(x) &= \{y | \text{ ancestor}(x, y)\} \\
L(x) &= \{y | \text{ ancestor}(y, x)\} \\
A(x, y) &= \{z | z \in H(y) \wedge z \notin \{x\} \cup H(x) \cup L(x)\} \\
T &= \{x | x = e \vee (x \in L(e) \wedge A(x, e) = \phi)\} \\
U &= \{x | x \in H(e) \vee (x \in L(e) \wedge A(x, e) \neq \phi)\}
\end{aligned}
$$

Evaluating functions H, L and A involves the recursive function ancestor. Determining the sets T and U involves evaluating $A(e, x)$ for all $x \in L(e)$. To determine T, U and F for a query with type 2 uncertainty involving domain element e:

1. Evaluate $H(e)$ and $L(e)$.
2. Evaluate $A(e, x) \forall x \in L(e)$.
3. Calculate $T = \{e\} \cup \{x | x \in L(x) \wedge A(e, x) = \phi\}$.
4. Calculate $U = H(e) \cup \{x | x \in L(x) \wedge A(e, x) \neq \phi\}$.
5. Calculate $F = \Omega - T - U$.

For a domain with only type 1 uncertainty, $A(e, x) = \phi$ for all $x \in L(e)$, which eliminates step 2 and simplifies steps 3 and 4. Algorithms whose complexity can be described in terms of the number of elements n in the domain have been developed for functions $H(e)$, $L(e)$ and $A(e, x)$ [8]. These algorithms are all polynomial – $O(n^4)$.

5 Relational Algebra Operations for Lattice-structured Domains

Introducing lattice-structured domains necessitates some extensions to the relational languages underpinning the relational model. Consider the five basic operations of the relational algebra: σ (selection), π (projection), $\times$ (cartesian product), $\cup$ (set union) and $-$ (set difference). Working with these domains will not affect π and $\times$, but will affect σ, $\cup$ and $-$. All other operations, such as $\cap$ (set intersection) and $\bowtie$ (join), can be constructed from the basic operations.

For simplicity, the relations in this section all contain exactly one attribute defined on a lattice-structured domain. The examples use the relations R and S (primary key `name`), and relation Q (primary key `(location, language)` defined in Table 2. As is usual for SQL, results of queries are allowed to include duplicate tuples. The changes required for σ are due to the effect of structured domains on conditions as previously described. This can be indicated in the algebra by using the inductive condition notation developed in the previous section. For example, $\sigma_{?C_2}(R)$ would return all tuples from R where C_2 is true or uncertain, i.e. {(Betty, Nicaragua), (John, Brazil)}.

The various join operations where join-columns are defined on lattice-structured domains can be expressed using inductive conditions in a similar way. Consider relation Q which shows the official languages for all six countries. The attribute `location` is defined on a flat subset of domain D which has been restricted to countries. $R \bowtie_{?(R.\text{location} = Q.\text{location})} Q$ returns all tuples from $R \times Q$ where the location values are or may be equal, whereas $R \bowtie_{R.\text{location} = Q.\text{location}} Q$ only includes the tuples where the location values are known to be equal. Table 6 shows some joins involving relations Q and R.

For the set operations, uncertainty can be introduced when relations with different levels of refinement for the same information are combined. Without further knowledge, it is reasonable to choose the information with the finest granularity when combining these relations, but sometimes this involves a loss of information. Susan has different locations in each relation, Chile and South America, only one of which can appear in $R \cup S$ and $R \cap S$. Chile is the most specific, so it is chosen. Peter on the other hand has the locations southern hemisphere and Brazil. Choosing either location causes the loss of some information, either the hemisphere, or the country. Without allowing attributes to take on multiple values, the best we can do is to make a decision about which information is the most useful. In cases such as this where one of the values is a descendent of the other in the lattice, the best method is likely to be to choose the descendent, and include or exclude the tuple depending on the set operation being performed. (Where neither of two different values representing the same information is the descendent of the other, they represent inconsistent information.) Table 7 shows $R \cup S$, $R \cap S$ and $R - S$.

Table 6. Joins involving relations R and Q

$R \bowtie_{R.\text{location} = Q.\text{location}} Q$

name	R.location	Q.location	language
Betty	Nicaragua	Nicaragua	Spanish
John	Brazil	Brazil	Portugese
Susan	Chile	Chile	Spanish

$R \bowtie_{?(R.\text{location} = Q.\text{location})} Q$

name	R.location	Q.location	language
Betty	Nicaragua	Nicaragua	Spanish
John	Brazil	Brazil	Portugese
Peter	southern hemisphere	Brazil	Portugese
Peter	southern hemisphere	Chile	Spanish
Susan	Chile	Chile	Spanish

$R \bowtie_{?(R.\text{location} = \text{'Brazil'})} Q$

name	R.location	Q.location	language
John	Brazil	Brazil	Portugese
Peter	southern hemisphere	Brazil	Portugese

$R \bowtie_{?(R.\text{location} = \text{'northern hemisphere'})} Q$

name	R.location	Q.location	language
Betty	Nicaragua	Nicaragua	Spanish
John	Brazil	Brazil	Portugese

6 Query Language Implications

Query languages such as SQL include statements for data definition and data update as well as querying the database. In this section describe an extension to the WHERE clause for inductive queries. See [8] for suggestions on defining lattice-structured domains and how the syntax of multi-table queries using the JOIN keyword might be adapted.

The WHERE clause needs to be adapted to incorporate inductive queries using the ? and ! symbols. The keyword MAYBE can be used as a condition modifier to represent ? with no confusion, since ? and $\neg$ are associative (i.e. $\neg(?C) = ?(\neg C)$). Since $!C = !(\neg C)$ care needs to be taken with choosing a keyword for !. TRUEORFALSE, although clumsy, has the advantage that the equivalence between TRUEORFALSE C and TRUEORFALSE NOT C is clear, and NOT TRUEORFALSE C can be used for $\neg(!C)$. UNCERTAIN C could be used instead of NOT TRUEORFALSE C, but this might lead to semantic confusion

Table 7. Results of set operations using R and S

$R \cup S$		$R \cap S$		$R - S$	
name	location	name	location	name	location
John	Brazil	Susan	Chile	John	Brazil
Susan	Chile	Peter	Brazil	Betty	Nicaragua
Peter	Brazil				
Betty	Nicaragua				

between UNCERTAIN and MAYBE. Using just MAYBE and TRUEORFALSE leads to WHERE clauses of the type shown below:

```
WHERE NOT TRUEORFALSE location = 'northern hemisphere'
WHERE MAYBE location = 'northern hemisphere'
WHERE MAYBE NOT location = 'northern hemisphere'
WHERE TRUEORFALSE location = 'northern hemisphere'
```

7 Conclusions and Further Research

Hierarchical domains are useful whenever there is a need to combine data from relations with different schemas, whether that difference arises from schema evolution over time in the same database or from relations from different sources. One important area of application is likely to be queries of web databases. Concept hierarchies used for web-based queries are discussed by Davulcu et al in [3].

Incomplete and imprecise information of the type that can be represented in concept and interval lattices can arise in many ways, such as by summarising or sampling data, from space restrictions imposed by mobile equipment or main memory databases, by needing to infer values for points not explicitly stored, and performance considerations in multimedia databases.

Allowing domain hierarchies to be lattice-structured rather than simple trees does not add much complexity to hierarchical domains: their definition and representation in the database is no more complex, and the additional (type 2) uncertainty introduced does not increase the complexity of the algorithm for determining the results of inductive queries.

Further work needs to be done to investigate the use of the lattices of intervals and attribute sets discussed in Section 2 of this paper, and how best to represent and reason with inconsistent information in databases. It is also important to test the behaviour of the hierarchical algorithms on realistic examples, and improve them where necessary.

References

1. P. Besnard and T.H. Schaub. Circumscribing inconsistency. In *Fifteenth International Joint Conference on Artificial Intelligence*, 1997.
2. D. Corbett and R. Woodbury. Unification over constraints in conceptual graphs. In W. Tepfenhat and W. Cyre, editors, *Seventh International Conference on Conceptual Structures*. Springer, 1999.
3. H. Davulcu, J. Freire, M. Kifer, and I.V. Ramakrishnan. A layered architecture for querying dynamic web content. In *ACM SIGMOD*, pages 491–502, Philadelphia, USA, 1999. ACM.
4. Z. Kedad and E. Metais. Dealing with semantic heterogeneity during data integration. In Jacky Akoka, Mokrane Bouzeghoub, Isabelle Comyn-Wattiau, and Elisabeth Metais, editors, *18th International Conference on Conceptual Modeling*, volume 1, pages 325–339, Paris, 1999. Springer.
5. H. Mannila. Inductive databases and condensed representations for data mining. In Jan Maluszynski, editor, *International Logic Programming Symposium*. MIT Press, 1997.
6. S. Parsons. Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):353–372, 1996.
7. D. Perlis. Sources of, and exploiting, inconsistency: preliminary report. *Applied Non-Classical Logics*, 7(1):13–24, 1997.
8. S.P. Rice and J.F. Roddick. Lattice-structured domains to represent imperfect data in databases. Technical Report ACR-00-006, University of South Australia, June 2000.
9. J.F. Roddick. *A Model for Temporal Inductive Inference and Schema Evolution in Relational Database Systems*. Doctor of philosophy, La Trobe University, 1994.
10. J.F. Roddick, N.G. Craske, and T.J. Richards. Handling discovered structure in database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):227–240, 1996.
11. J.F. Roddick and S.P. Rice. Towards inductive queries. In *Ninth Australasian Conference on Information Systems*, volume 2, pages 534–542, Sydney, Australia, 1998. University of New South Wales.
12. P. Smets. Probability, possibility and belief: which and where? In Dov M. Gabbay and Philippe Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Quantified Representation of Uncertainty and Imprecision*, volume 1, pages 1–24. Kluwer Academic Publishers, 1998.

Preprocessing of Requirements Specification

Kroha,P.

Technical University of Chemnitz,
Department of Informatics, 09107 Chemnitz, Germany

Abstract. In this paper, we discuss the part of the requirements specification process which is located between the textual requirements definition and the semi-formal diagrams of the requirements specification. It concerns the acquisition and the refinement of requirements and the consensus improvement between the analyst and the user. We argue that the existence of a textual description of requirements that represents the analyst's understanding of the problem will improve the efficiency of the requirements validation by the user. A method and a tool are introduced that support the acquisition, refining, and modeling of requirements, and finally the automatic generation of a textual description of the model for the validation.

1 Introduction

To get a contract for a large software project, the software house has to present a requirements definition. Its purpose is to describe all features of the new system that are necessary for the customer to sign the contract. Using a natural language is necessary because a customer would not sign a requirements definition written e.g. in the Z notation.

Once a contract has been awarded and after a feasibility study has been approved, a requirements specification must be written that describes the functionality and constraints of the new system in a more detailed way. It will usually be written in some semi-formal graphical representation given by the used CASE tool. Since software engineers are not specialists in the problem domain, their understanding of the problem is immensely difficult, especially if routine experiences cannot be used. It is a known fact [2] that projects completed by the largest software companies implement only about 42 % of the originally-proposed features and functions.

We argue that there is a gap between the requirements definition in a natural language and requirements specification in some semi-formal graphical representation. The analyst's and the user's understanding of the problem are usually more or less different when the project starts. The step from the requirements definition towards requirements specification will be done only in the brain of the analyst without being documented. Usually, the user cannot deeply follow the requirements specification. The first possible time point when the user can validate the analyst's understanding of the problem is when a prototype starts to be used and tested.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 675–684, 2000.

We offer a textual refinement of the requirements definition which can be called requirements description. Working with it, the analyst is forced by the supporting tool TESSI to complete and explain requirements and to specify the roles of words in the text in the sense of the object-oriented analysis. During this process, a UML model will automatically be built by TESSI driven by the analyst's decisions. This model will be used for the synthesis of a text that describes the analyst's understanding of the problem, i.e. a new, model-derived requirements description will automatically be generated. Now, the user has a good chance to read it, understand it and validate it. His/her righting comments will be used by the analyst for a new version of the requirements description. The process repeats until there is a consensus between the analyst and the user. This does not mean that the requirements description is perfect, but some mistakes and misunderstandings are removed.

We argue that the textual requirements description and its preprocessing by TESSI will positively impact the quality and the costs of the developed software systems because it inserts additional feedbacks in the development process.

The rest of the paper is organized as follows. In section 2, we discuss the related work. In section 3, the presented method will be described in detail. In section 4, we mention the XML-interface to the tool Rational Rose. Section 5 describes advantages of our method for the software development. The last section discusses the implementation and conclusions.

2 Related work

There are many good books on requirements engineering, e.g. [15]. Specialized papers related to our topic try either to filter some semantic information from a text (analysis) or to generate text from some diagrams (synthesis). We have not found any papers that combine these concepts to build a feedback in requirements specification.

To obtain semantic information directly from a text, manual or automatic methods can be used. There are many papers and books [1] that support manual methods. For example, the method of grammatical inspection of a text can be mentioned. It analyzes the narrative text of use cases, identifies and classifies the participating objects [5]. In this method, nouns will be held for candidates for objects or classes, adjectives will be held for candidates for attributes, and verbs will be held for candidates for methods or relationships. The Object Behaviour Analysis [16] belongs to this group of methods, too. Nevertheless, we have not had any opportunity to use a CASE tool which supports these methods.

There are some tools helping in requirements management (DOORS, Requisite Pro) that transform unstructured texts of requirements into structured texts. They neither build any models nor generate refining questions like our tool TESSI.

Automatically generated natural-language description of software models is not a new idea. The first system of this kind was described in [18]. More recent projects include the ARIES [6] and the GEMA data-flow diagram describer [17].

Even if it is widely believed that the graphical representation is the best base for the communication between the analyst and the user, there are some serious doubts about it. It has been found in [7] that even a co-operation between experienced analysts and sophisticated users who are not familiar with the particular graphical language (which is very often the case) results in semantic error rates of about 25 % for entities and 70 % for relations. Similar results brings [14]. Some systems have been built which transform diagrams, e.g. ER-diagrams, into the form of a fluent English text. For example, a system MODEX has been described in [12], [13] which can be used for this purpose. Differently to our system, input to MODEX (MODel EXplainer) is based on the ODL standard from the ODMG group.

The first version of our system TESSI has been introduced in [10]. Additionally to MODEX, it supports the manual analysis of the requirements description by the analyst and the semi-automatic building of an OO-model in UML. Differently from MODEX, we have used simpler templates for generating the output text, because our focus was not in linguistics. The current version of TESSI generates refining questions and uses an XML-interface to Rational Rose.

3 Preprocessing of Requirements

The acquisition of requirements is a very important part of software engineering. It mostly consists of interviews with users about the problem domain and their use cases. We argue that

- this process should be supported by tools,
- the best way to represent the obtained information at this level of knowledge is not a diagram as commonly accepted but a text.

After the first version of requirements description has been written by the analyst, it will be analyzed by him/her (supported by our tool TESSI - TExtual aSSIstent) to build an object-oriented model and to validate it in a co-operation with a user.
This process can be divided into:

- model identification
  - identification of static features supported by generated questions,
  - identification of dynamic features supported by generated questions,
- model validation
  - automatic generation of a text that corresponds to the identified object-oriented model,
  - validation of this generated text by the user,
  - revision and correction of the last version of the requirements description by the analyst,
  - evaluation whether the whole process should be repeated.

During the model identification that is based on the grammatical inspection, TESSI generates complementary questions to force the analyst not to forget to describe some specific features of the system. The validation of requirements supported by the automatic generation of the model-derived requirements description is the next step [10]. The generated requirements description represents the analyst's understanding of the problem.

We argue that the comparison of the analyst's understanding and the user's understanding provides an important feedback. The generated text will be read by both the user and the analyst. It will be compared with the original requirements description in order to approve it or to correct/complement it. The corrected original requirements description (resp. its corrected and newly appended parts) will be then analyzed again by using the same method.

This process will be repeated while the user and the analyst see any lacks and suspicious formulations in the generated requirements description. After consensus, the current UML model represents the starting point for development of the functional requirements specification and the prototype.

3.1 Identification of the model

The analyst's decision about the role of words in the textual requirements description is one of the focal points of the presented method and it is supported by interface menus of TESSI. We do not believe that such a decision can automatically be made without a human interaction. The semantic vagueness of natural languages and the context dependence are too hard problems.

TESSI supports object-oriented modeling in UML. For modeling static features, we use class diagrams and the concepts like class attribute, class operation, class cardinality, generalization, association, and aggregation. For modeling dynamic features, statechart diagrams and sequence diagrams have been supported until now. In a requirements description, a word can be irrelevant, it may denote a candidate for an object, a class, an attribute, a relation, a method, a constraint [1], [10].

Each decision of the analyst causes changes (usually APPEND) in internal data structures representing the model. They have bi-directional pointers to the corresponding source text segments and to templates for generating refining questions (see section 3.2).

The process of identification of dynamic features is based in principle on Object Behaviour Analysis [16].

3.2 Questions and their templates

Observing, asking questions and looking for answers is in principle the only method how to investigate. In common, it is a creative activity, but in a specific scope of requirements acquisition, there are many known standard questions that can mechanically be asked and that must be answered to discover an essential subset of classes, their attributes, methods, relations, etc. We distinguish two kinds of questions that differ in how the answer is intended to be used.

Expanding questions will be used when writing requirements, i.e. at the begin of the iterative process. At this level of knowledge we hope to find new relevant words in answers that would help us to identify new entities that are not contained in the existing text. We have used templates like "What is the purpose of <placeholder> in the system?", "What is the role of <placeholder> in the system?", "What is the responsibility of <placeholder> in the system?". These questions will be triggered after the context of a sentence is left for classes identified in the sentence. The analyst can ignore the question if he/she can find the answer in the next text.

Example:
Starting text: An information system for a library shall be written.
The analyst identifies LIBRARY as a candidate for a class and will be asked: "What is the purpose of the LIBRARY in the system?" There are many possibilities how to answer. An experienced analyst answers, for example, "In a LIBRARY librarians lend books to readers." Even if this is an answer at a children level it helps to refine the system. An unexperienced analyst could answer: "A LIBRARY improves the cultural level of the nation." This is not completely wrong, but it is absolutely useless for requirements acquisition.

(End of example)

Completing questions will be generated to complete the existing text of requirements description. As a last step of the identification process, TESSI checks whether all attributes of the internal modeling data structures containing the object-oriented model are completed. Every not completed attribute triggers a question generation. Because of the bi-directional pointers, the context of such a question can be found and shown. The analyst checks the reason, inserts (usually) an additional sentence (or more) to the requirements description that obtains the missing information, and completes the model.

Of course, the generated questions do not guarantee any completeness of the model, because even if TESSI generates many questions, it cannot check the correctness of the answers. This means that the generated questions remind the analyst on some, maybe forgotten, modeling features.

As far as the implementation is concerned, the question templates build a set of fixed pre-built questions that are parameterized by names taken from the associated data modeling structures. Templates for generating questions and templates for generating the text description of the model are associated with internal data structures in which TESSI stores the identified entities. Templates are strings with placeholders in the current implementation of TESSI. Before generating, placeholders will be substituted by names of entities described in the associated data structures.

In MODEX [13], templates are complex structures representing not only the contents of the sentence but also its grammatical structure in English.

3.3 Transformation of the model into a structured text

After the analyst's processing of the requirements description has been finished, there is a complete UML model available in the internal data structures. TESSI can automatically generate the textual description of this model using a one-to-one mapping between entities of the model and textual templates. This generated version of the requirements is equivalent to the UML model and describes how the analyst understands the problem. Denoting the UML model as a component of requirements specification, the generated text can be denoted as a textual description of this component of requirements specification.

The text generation starts at the root (roots) of the class hierarchy. After the structure of the class hierarchy has been used for the text generation, the description of specific classes and their static features will continue. The description of each statechart diagram will be generated as a part of a class description, the description of sequence diagrams will be generated separately as use cases.

The generated text sounds sometimes unusual, but it contains all features important for understanding. All words which were not marked in the original text will be held for irrelevant and will be omitted in the generated text. New irrelevant words come from the used templates.

A set of templates can be constructed for any natural language. It is also possible to define pairs of templates in two (or more) natural languages and generate the requirements description in more than one language. In this case, names of entities (classes, attributes, etc.) should be international because they will not be changed by the templates.

Some linguistic problems occurred that concern the complexity of different natural languages, e.g. irregular plurals, suffixes of adjectives, articles, etc. in the German language. The language-dependent parts have been programmed in separated units. Our focus is not in linguistics, so we did not try to achieve a linguistic perfection.

The generated text will be used for validation of the requirements description by the user as stated above.

3.4 Feedbacks in the requirements description

The first feedback will be provided during the identification of static and dynamic features of the UML model. Each identified concept has a corresponding list of question templates that forces the analyst to complete the necessary details, e.g. cardinality of relations, constraints of attributes, etc.. As a result the analyst inserts some sentences into the text of requirements as an answer to the generated question.

The next feedback is represented by the validation of the generated text by the user. In co-operation with the analyst, the user compares the requirements description, his/her knowledge, and the generated text. Discrepancies will be discussed, noticed, and some corrections will be done in the last version of requirements description.

Example:

A sentence in a requirements description: "A table has legs." The analyst recognizes that "table" and "leg" can be candidates for classes. An experienced analyst can recognize that between the class "TABLE" and the class "LEG" could be an aggregation relation "Consists-Of". If he/she does not recognize it, TESSI generates a question: "Is there some relationship between TABLE and LEG?". Such questions will always be generated for names of classes that are included in one sentence without having an identified relationship.

After the analyst identifies the relationship between TABLE and LEG as "Consists-Of", TESSI generates a question about the cardinality of the identified relationship "How many LEGs does a TABLE consists of?". Now, the analyst either knows it or asks the user. Then he/she inserts a sentence "A table consists of four legs." into the text of requirements specification. In this sentence, "four" will be identified as a cardinality "1 to 4" of the relationship "Consists-Of" between the classes TABLE and LEG. It will be noted in internal structures of TESSI that describe the model.

The validation by a user represents the second feedback. This time the user (accompanied by the analyst) compares sentences: "A table has legs" in the original text of requirements description with "A table consists of four legs" in the generated text. The result of this comparison may be that the user recognizes that this description is not complete because a table has also some other parts, e.g. a desk. However, it may be that the existence of a desk is not relevant. Suppose that it is relevant and that the analyst has again forgotten to describe the cardinality and that the new, improved version of the text will be: "A table consists of four legs and a desk." The analysis of this sentence signals an existence of a class DESK. If the analyst does not identify a relationship "Consists-Of" between TABLE and DESK then questions follow: "Is there some relationship between TABLE and DESK?", "How many DESKs does a TABLE consists of?", "Is there some relationship between LEG and DESK?". The last question does not bring anything in this case but it will mechanically be generated, because TESSI does not understand semantics of our world. In this way the iterative process of requirements description develops.

(End of example)

4 XML-Interface to Rational Rose

We did not intend to substitute any professional CASE tool. Our motivation was to document the phase of the analyst's thinking during requirements acquisition and modeling. TESSI helps only during the preprocessing of the requirements description and builds a model. It can generate a skeleton of a program corresponding to the UML model in C++ and POET, but it does not draw any diagrams.

As the next step in requirements specification we transfer the validated UML model in XML from TESSI to Rational Rose to design a prototype using all possibilities of Rational Rose.

Currently, an interface to Rational Rose has been implemented [4]. There are some pre-built extensibility possibilities in Rational Rose 4.0 available. The model built by Rational Rose can be accessed through the RREI (Rational Rose Extensibility Interface) using a script language. For Rational Rose, a script has been written that generates (for Rational Rose output) or interprets (for Rational Rose input) the UML model in XML format. In TESSI, the XML format description of the UML model will be generated or accepted, too. This makes the exchange of models between TESSI and Rational Rose possible. The interpretation of the XML format in TESSI has been appended as a separate phase. This interface makes it possible to send the TESSI-model that represents the validated UML model to Rational Rose to draw diagrams and to build a prototype. It is also possible to send UML diagrams (including all corrections) from Rational Rose back to TESSI and to generate the corresponding textual description.

5 Advantages for the software development

The preprocessing of requirements not only improves the quality of the analysis (requirements will better be understood) and the quality of the design (main program structures will automatically be created during the analysis and are directly bound to entities of the analysis). All aspects of the software system evolution (adaptive maintenance, new black-box test cases, etc.) will also profit when using the method described above.

5.1 Metrics as a side effect

At the very beginning of every project, the customer asks about the costs and delivery time. The preprocessing of requirements brings the possibility to use metrics at this phase of the development. Currently, the most simple object-oriented metric has been used in TESSI. Corresponding to every version of the requirements description, the analyst knows how many classes, methods, attributes, relationships, etc. will probably be used in the description of the system. The costs of the product development, its time schedule, the volume of tests, personal resources, etc. can be derived from this knowledge if enough experience has been collected during similar projects. Additionally, the possible impact of every change in the requirements can be followed during the discussion with the user.

5.2 Traceability of requirements

In software evolution and adaptive maintenance, traceability of requirements is very important. When we move from analysis to design, we assign requirements to the design elements that will satisfy them. When we test and integrate code, our concern is with traceability to determine which requirements led to each piece of code. The tool TESSI builds and maintains some important relationships between requirements and parts of the resulted system, e.g. bi-directional

links between the identified entities in sentences of textual requirements and the corresponding entities of the model. These links will be followed during an adaptive maintenance. This helps to hold requirements and programs consistent and supports the concept of software evolution in which every change in the software system should start with the change of the requirements specification and follow the life-cycle of development.

6 Implementation and Conclusions

The first prototype without generation of questions was implemented in Turbo-Pascal and Turbo-Vision [10], the improved version in Delphi later on. Except of simple examples (50 - 100 sentences) TESSI has been used for processing requirements of its own which contained several hundreds of sentences. Using TESSI in practice we proved the very known facts:

- Textual requirements description needs some experience.
- Use cases should be organized into groups according to their actors with allocated functional requirements.
- Several iterations are necessary.
- In the generated text, attributes and methods are described, but not the purpose.
- The most important parts of the problem's semantics are usually not included in the first version of the requirements. The reason is that the software analyst doesn't know about their existence and the customer holds them for so much self-evident that he/she doesn't mention them at all.

The current version of TESSI has been implemented in Java [11] and includes the XML-interface between TESSI and Rational Rose 4.0 [4]. This combination of tools offers some very promising possibilities that will be used and tested in the future. The current version has just been finished, so that there was not enough time to get, collect, and evaluate some experience. Currently, we are looking for a co-operation in a project where TESSI will be used in parallel with classical technology in the sense that one group of analysts will directly use Rational Rose and the other group will use at first a preprocessing via TESSI. The goal will be to compare the necessary effort and the quality achieved in both cases.

References

1. Coad, P., Yourdon,E.: Object-Oriented Analysis. Prentice Hall, 1991.
2. CHAOS Report, The Standish Group, 1995.
3. Emmerich, W.,Kroha, P., Schäfer, W.: Object-Oriented Database Management Systems for Construction of CASE Environments. In: Marik, V. et al (Eds.): Proceedings of the 4th International Conference DEXA'93, Lecture Notes in Computer Sciences, No. 720, Springer, 1993.
4. Gemeinhardt, L.: Connecting TESSI and Rational Rose by means of XML. Project Report, TU Chemnitz, 2000. (In German)

5. Jacobson, I.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992.
6. Johnson, W.L., Feather, M.S., Harris, D.R.: Representation and presentation of requirements knowledge. IEEE Transactions on Software Engineering, pp. 853-869, 1992.
7. Kim, Y.-G.: Effects of Conceptual Modeling Formalism on User Validation and Analyst Modeling of Information Requirements. PhD Thesis, University of Minnesota, 1990.
8. Kroha, P.: Objects and Databases. McGraw-Hill, 1993.
9. Kroha, P.: Softwaretechnologie. Prentice Hall, 1997 (in German).
10. Kroha, P., Strauß, M.: Requirements Specification Iteratively Combined with Reverse Engineering. In: Plasil, F., Jeffery, K.G. (Eds.): Proceedings of SOFSEM'97: Theory and Practice of Informatics, Lecture Notes in Computer Science, No. 1338, Springer, 1997.
11. Leidenfrost, S.: TESSI in Java. Project report, TU Chemnitz, 1999. (In German)
12. Lavoie, B., Rambow, O., Reiter,E.: The ModelExplainer. In: Demonstration Notes of International Natural Language Generation Workshop, INLG'96, Harmonceux Castle, Sussex, UK, 1996.
13. Lavoie,B., Rambow, O., Reiter, E.: A Fast and Portable Realizer for ext Generation Systems. In: Proceedings of the 5th Conference on Applied Natural Language Processing, ANLP'97, Washington, 1997.
14. Petre, M.: Why looking isn't always seeing: Readership skills and graphical programming. Communication of the ACM, Vol. 38, No. 6, pp. 33-42, 1995.
15. Robertson, S., Robertson, J.: Mastering the Requirements Process. Addison-Wesley, 1999.
16. Rubin,K., Goldberg,A.: Object Behaviour Analysis. Communication of ACM, Vol. 35, No.9, pp. 48-62, September 1992.
17. Scott, D., de Souza, C.: Conciliatory planning for extended descriptive texts. Technical Report 2822, Philips Research Laboratory, Redhill.
18. Swartout,B.: GIST English Generator. In: Proceedings of the National Conference on Artificial Intelligence, AAAI'82.

Analyzing Enterprises: The Value Cycle Approach

P.R. Griffioen[1], P.I. Elsas[2], and R.P. van de Riet[3]

[1] Free University of Amsterdam and Deloitte & Touche Bakkenist
pgriffioen@bakkenist.nl
[2] Free University of Amsterdam and Deloitte & Touche Bakkenist
elsas@cs.vu.nl
[3] Free University of Amsterdam
vdriet@cs.vu.nl

Abstract. In this paper we consider a formalism for studying specifications of the structure and behaviour of enterprises. An enterprise is an organisational system producing goods or services for the market in order to make financial profit. An important issue is the value cycle, used for determining the entrepreneurial viability. The formalism offers the advantage of rigorous automatic analysis in design time of central properties of the enterprise specification. The formalism uses so-called value cycle nets, based on Petri nets.

1 Introduction

Value cycles have been used in audit theory for years [5], [8] and [9]. The concept of a value cycle lends itself very well for formalisation which in turn allows for rigorous analysis. In [2] such a formalism is developed and used to build a model for mathematical audit. Central to the theory are the concepts of value reachability and conspiracy computation. The last one is further developed in [3]. Audit deals with the past behaviour of an enterprise but values cycles can also be applied to well-founded analysis of future behaviour.

In the emerging discipline of Enterprise Engineering an enterprise is viewed as a complex system of processes that can be engineered to accomplish specific organisational objectives. Enterprise models assist the goal of Enterprise Engineering by helping to represent and analyse the structure of activities and their interactions [6]. In this paper we consider models based on the value cycle of an enterprise. With the models what-if scenarios can be analysed with respect to the effect on the overall behaviour of enterprises (e.g. prognosis of turnover).

A value cycle is a holistic processual framework for business notions. A consequence of the formal approach is that business notions can be given a strict meaning. For example the notion profit can be defined as the revenues obtained by the sales process minus costs made in all processes in the value cycle. Because a value cycle is expressed in business notions, an understanding by business people is possible. A value cycle is holistic because it is a framework in which all processes of an enterprise are related. This property makes analysis of the overall behaviour of an enterprise, e.g. profitability, possible.

This paper starts with an informal description of value cycles. Next we show how the value cycle is formalised and give a state-vector interpretation to support compu-

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 685-697, 2000.

tation. After that we introduce the basic model of a value cycle and extend it with deontic aspects. To explain how different types of enterprises can be modelled we show the value cycles of the major types of the enterprises from the typology of Starreveld. We conclude with a case study of a project in which a value cycle has been used to build an analysis tool for an enterprise.

2 Value Cycle

An enterprise is an organisation which produces for the market with the aim of making profit. The enterprise's circular flow of business values, or value cycle for short, is a key notion in Dutch auditing theory [1]. It is the cyclic interrelationship between all primary processes of an enterprise. An example is the value cycle of a trading enterprise which is shown in figure 1 [10].

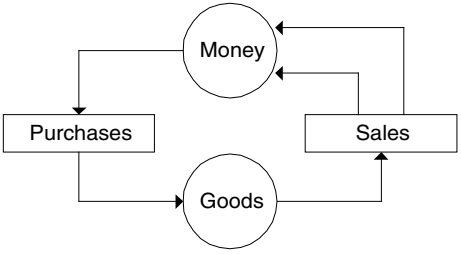


Fig. 1. A value cycle of en elementary trading enterprise. This enterprise uses money to purchase goods and at a later moment sells these goods to obtain money. The money and the goods are the subtypes of value that are transformed by the transactions purchases and sales

To make a clear distinction between the structure and the behaviour of value cycles, models like figure 1 are called structural value cycles and a 'tour' in a structural value cycle is called a value cycle. Generally, a value cycle involves all transactions in the structural value cycle to occur zero or more times. The structural value cycle imposes constraints on the actual number of times each transaction is to occur and also on the order in which they are to occur. The enterprise in figure 1 for example, cannot sell more than it has bought previously. Since an enterprise produces for the market to make financial profit, actions in the value cycle are to lead to a positive value difference. In the trading enterprise, this is the difference between the sales price and the purchase price. The result of the occurrence of a value cycle (i.e., 'value after' minus 'value before') is called a value jump, or added value, or alternatively, gross margin per product. In the example of the elementary trading enterprise the purchase and the sales transactions both interact with the environment of the company. Goods flow into the enterprise at the purchase transaction and flow out of the enterprise at the sales transaction while money flows into the enterprise at the sales transaction and flows out of the enterprise at the purchase transaction.

3 Formalism

Structural value cycles are specified in terms of enriched Petri nets. Petri nets are a formal model for true concurrent processes. In the enriched Petri nets a marker represents a unit of (registered) business value. A place represents a marker's buffer location. A transition represents a transformation of business values. From the basic Petri net elements we construct buffers and transactions as shown in figure 2.

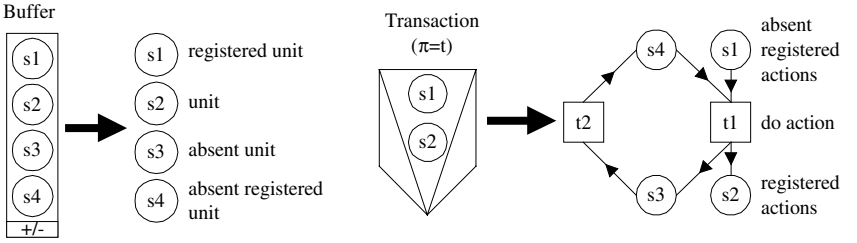


Fig. 2. Buffer type and Transaction type: A buffer is built from four places and has a sign attached to it modelling whether or not the values in the buffer are an asset. The outermost two places model the registration of units and the innermost two places model the business values themselves. The two places at the side of the sign represent absence and correspond to the other two representing presence. The absence and presence places together model the fixed capacity of values and their registration. The transition labelled 'do action' in a transaction models an action of an enterprise. The buffer 'registered actions' registers the occurrences of all the transitions labelled 'do action', consuming free registration space from 'absent registered actions'

Every transaction has to occur zero or more times in a value cycle. This number is called the proration value and is notated as the symbol π in the label of the transaction. Figure 3 shows the occurrence of a transaction.

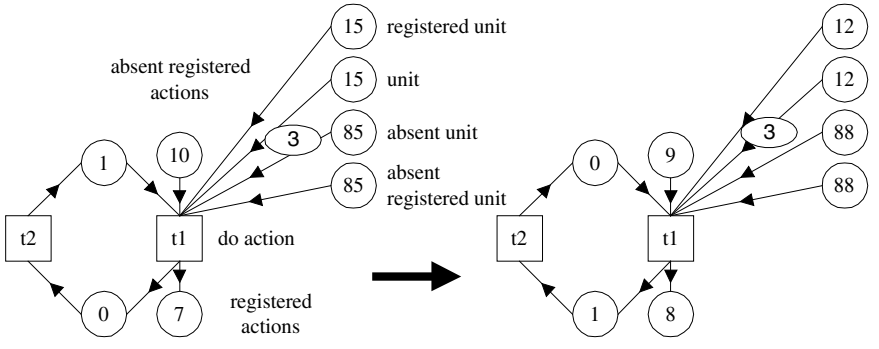


Fig. 3. A transaction occurrence in a fragment of a value cycle. The integers in the places denote the number of markers contained in the places

Structural value cycles are built by interconnecting buffers and transactions in an alternating way. The places in a buffer attach to the 'do action' transition. The cardinality of the connection to a presence place must be equal to the cardinality of the

connection to the corresponding absence place, while the direction of these connections must be opposite. The graph of connected buffers and transactions must consist of one component. To enforce the requirements structural value cycles are constructed from start configurations and building rules. Properties can be derived from the net from the principles followed in its construction [7].

3.1 Basic Model: Trade Elementary

A formal model for the elementary trading enterprise from figure 1 is built from the buffers 'Money' and 'Goods' and the transactions 'Sales' and 'Purchases'.

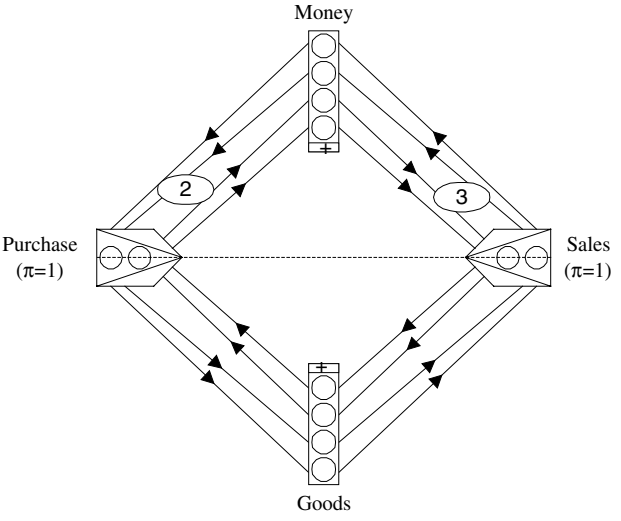


Fig. 4 . An Elementary Trading Enterprise: It uses 'Money' to purchase 'Goods' which are sold at a later moment which results in 'Money' again. The horizontal dotted line shows how the transactions divide the cycle in the flow of money and the flow of goods. The integers on the arrows denote the cardinality of the connection. It defaults to one if omitted.

3.2 Imputations

Every transaction occurrence consumes resources like personnel time or machine time per activity. To model the resource usage there are so-called impute transactions that consume money and produce units of a certain resource type. The amount of money consumed is the unit-price of the resource.

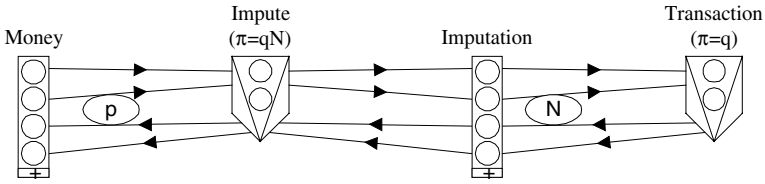


Fig. 5. Imputation

Impute transactions are equivalent to purchase transactions. They have been left from most figures in this article for reasons of brevity.

3.3 State Vectors

A Petri net state can be represented by a state vector. In paragraph 2 the result of the occurrence of a value cycle was called a value jump. To formalise the value jump we calculate the state vector that is the result of one value cycle occurrence.

Let B and T be the sets of buffers and transactions respectively. Now consider the underlying Petri net restricted to the set T' containing only the 'do action' transitions (t1 in figure 2), and restricted to the set B' containing only the presence places for value (s2 in figure 2). The flow matrix G for this pure Petri net is a matrix $B' \times T' \to I\!N$. Associated with each transaction $t \in T$ is its proration value $\pi(t) \in I\!N$. For transaction $t \in T$ and associated transition $t' \in T'$, the vector $\lambda \in I\!N^{T'}$ with $\lambda[t'] = \pi(t)$ is called the *proration vector*. The state vector $\upsilon \in I\!N^{B'}$ in the equation $G \circ \lambda = \upsilon$ is the result of the occurrence of one value cycle. Let P_{Sales} be the sales price, P_t the purchase price of a purchased good and Imp_t the unit-price of a resource. Using the properties of a structural value cycles it can be shown that $\upsilon[b] = J$ if b is the presence money place and zero otherwise with $J \in I\!N^+$ as follows:

$$J = P_{Sales} - \sum_{t \in T':Pur(t)} \pi(t) P_t - \sum_{t \in T':Imp(t)} \pi(t) Imp_t \qquad (1)$$

This value J is the jump value. It is the increase of money that results form a value cycle occurrence. The matrix equation $G \circ \lambda = \upsilon$ expresses the relation between the prorational activities, i.e. transition firing, and obtained results, i.e. the jumpvalue.

3.4 Deontic Aspects: Trade Extended

Direct transfer of values between enterprises is often not practical and therefore replaced by the mutual acceptance of the obligation to transfer values. From the viewpoint of one enterprise this creates simultaneously an obligation to delivery and a right to receipt of value. At the same time for the enterprise dealt with, the opposite deontic rights and obligations are created. When the actual transfer of values takes

place the corresponding obligation and right nullifies. In general a value cycle is the processual framework for into-existence-coming and out-of-existence-going of 'obligations to delivery' and 'rights to receipt' of values [10].

To include the deontic refinements, the elementary trading enterprise is extended with four buffers and four transactions. The obligation to delivery and the right to receipt of money are the buffers 'Creditors' and 'Debtors', respectively. For goods these buffers are 'Sales orders' and 'Purchase orders', respectively. Figure 6 shows the extended trading enterprise.

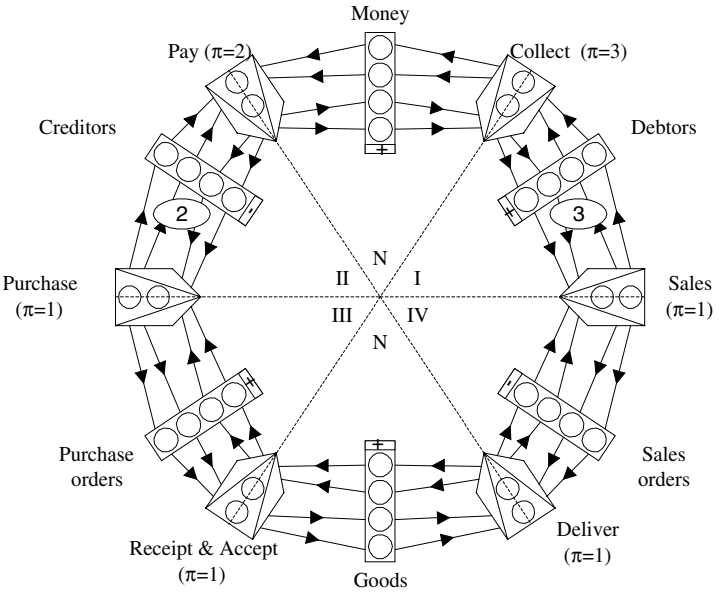


Fig. 6. Extended trading enterprise. The model is derived from the elementary trading enterprise with the so called stretch-rule or s-rule [2]. The s-rule connects a new buffer and connected transaction in the cycle to 'stretch' it Each part of the cycle is labelled with a deontic status from table 1.

Table 1. Deontic labels

Label	Deontic status	Enterprise dealt with
N	Neutral	Neutral
I	Right to receipt of money	Obligation to delivery of money
II	Obligation to delivery of money	Right to receipt of money
III	Right to receipt of goods	Obligation to delivery of goods
IV	Obligation to delivery of goods	Right to receipt of goods

Classification					Examples
organizations producing for the market	organizations with a dominant flow of own goods	organizations without technical transformation process	trade organizations delivering mainly to other industries		wholesalers, importers, exporters
			trade organizations delivering mainly to final consumers		shops, retailers
		industrial organizations	industrial organizations with homogeneous mass production	(flowingly) rotating homogeneous mass production	gas-works, power stations, sugar-factories, oil refineries, paper-mills
				parcelling homogeneous mass production	brick-works, breweries, tanneries, lime-kilns, wire-drawing mills
			industrial organizations with heterogeneous mass production	singular heterogeneous mass production	glass-works, potteries, wall-paper factories, preserving factories
				compound heterogeneous mass production	factories for: shoes, ready-made clothing, audio devices, bicycles, cars
			industrial organizations with (serial) piece production	(unique) piece production	cloth tailoring, house building con-tractors, shipyards, engineering works
				serial piece production	builders of: sisterships, ship mo-tors, railroad passenger cars
		agrarian and extractive organizations			agriculture, animal husbandry, horti-culture, forestry, mining industry, fishing industry
	organizations and professions without a dominant flow of own goods	service organizations	service organizations with flow of goods	some flow of goods owned by the organization	pubs, coffeehouses, restaurants, publishers of newspapers
				flow of goods owned by others	auctioneers, laundries, dye-works, repair-works, transporters, storehouses
				delivery of goods via fixed pipes or wires (is: outflow)	gas, electricity and water suppliers, telephone exploiters, radio and tv broadcasters
			service organizations offering space-time capacity	specific reservation of space-time capacity	house exploiters, hospitals, hotels, storehouses, transporters of passengers over relatively long distance
				unspecific reservation of space-time capacity	entertainment providers, swimming pools, theaters, transporters of passengers over relatively short distance
			other service organizations and professions		professional services, cleanup services
		financial institutions	banks		general banks, savings banks, mortgage banks
			special finance institutions		venture capital companies, investment companies (trusts)
			intermediates in stock exchange		stockbrokers
			insurance organizations		life insurance and indemnity insurance companies
organizations producing (and/or offering services) directly for their members, i.e. without mediation of the market			governmental agencies and public corporate bodies (as excluded above)		government, public corporate bodies (possibly belonging to organizations which produce for the market)
			private corporate bodies (as excluded above)		foundations, societies, religious communities

Fig. 7. Starreveld's typology

4 Typology

Starreveld made a typology of enterprises based on the commonality in their structural value cycles, see figure 7 [9]. In the following sections we sketch how different types of enterprises from the typology of Starreveld can be constructed with our formalism. Starreveld divides organisations that produce for the market into enterprises with a dominant flow of own goods and organisations and professions without a dominant flow of own goods. The trading enterprises of the first type were already discussed. In the next sections we discuss industrial enterprises and service enterprises.

4.1 Industrial Organisations

Industrial organisations have a technical transformation process that transforms purchased goods into goods to be sold. Figure 8 is an example of a bicycle manufacturer.

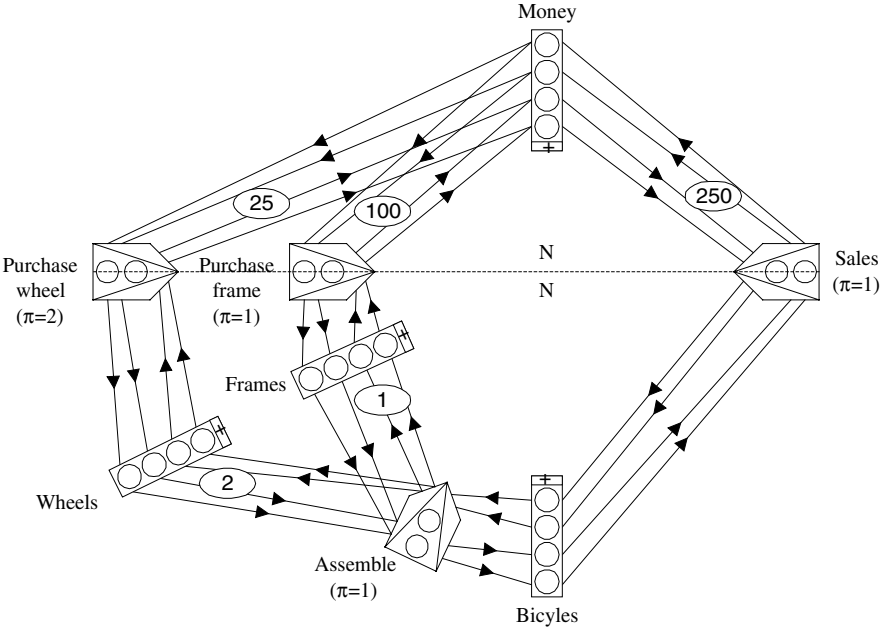


Fig. 8. A bicycle manufacturer who purchases wheels and frames and assembles them into bicycles. A bicycle consists of one frame and two wheels (for simplicity we ignore all other parts). This 'proportion' matches the proration values of the purchase transactions and the cardinalities of the presence consuming connections to the assemble transaction.

In general a transformation process can be put into the cycle with the proportion- or p-rule. The p-rule adds purchase transactions and buffers for all the goods required. It also adds a transaction that represents the technical transformation. The proration values for the purchase transactions must be equal to the cardinality of the connection between the buffers and the assemble transaction.

4.2 Service Enterprises

Pure service enterprises have no flow of own goods. Instead of selling products, the inflow of money is generated by the performance of services for customers who pay for it. Figure 9 shows the general structural value cycle of pure service enterprises.

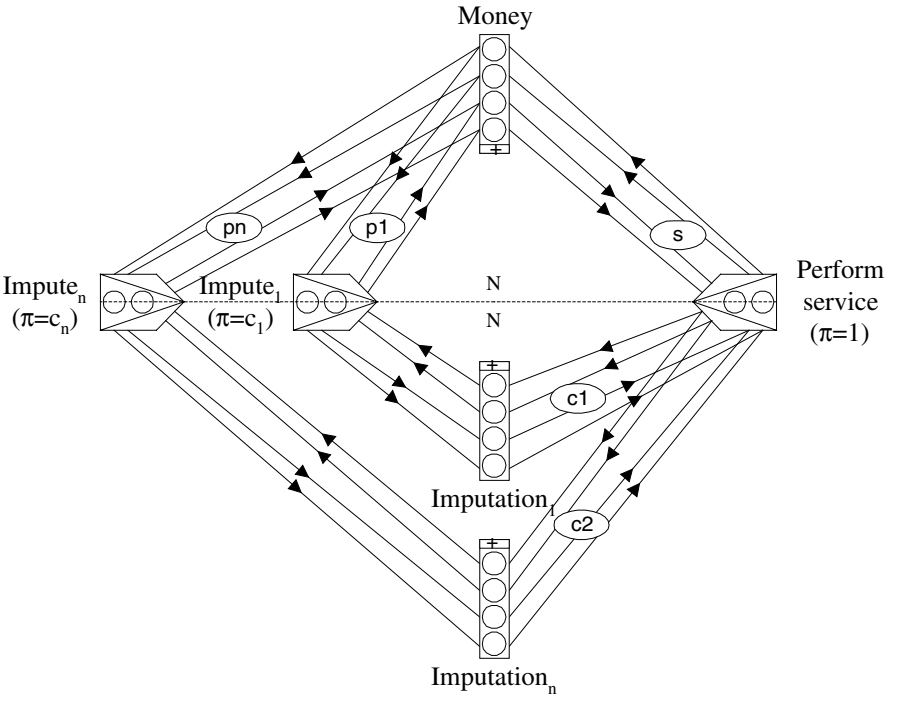


Fig. 9. A pure service organization. It performs services for which it receives money, consuming from various imputations

Some organisations are a combination of a pure service organisation and a non-service organisation. For example a restaurant produces meals and provides tables as a service to consume the meal. It is an example of the some flow of goods owned by the organisation from the typology of Starreveld. Figure 10 shows the structural value cycle of a simple restaurant that sells fish and chips only. It purchases 'fish' and 'chips' which are used by the 'cook' transaction. Cooking requires 'working time'. A 'table' is provided to a customer when a meal is served.

The different types of service enterprises can be constructed with the p-rule. The p-rule introduces zero or more purchase transactions and goods buffers. For a pure service enterprise the p-rule is applied with argument zero. For other service enterprises one or more products are introduced.

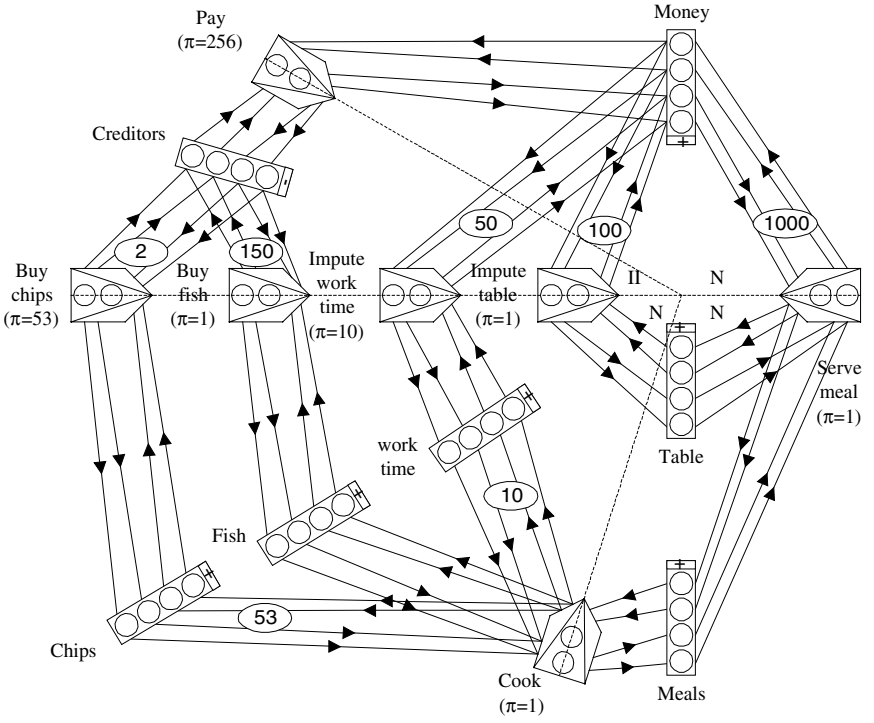


Fig. 10. A restaurant

5 Application

The concept of a structural value cycle is used to build an Activity Based Costing and Budgeting system for a large logistic enterprise. In the typology of Starreveld it is classified as a service organisation with a flow of goods owned by others. Figure 11 shows its structural value cycle. The transactions in it are 'Sales', 'Perform logistic service' and two impute transactions 'Impute cars' and 'Impute personnel'. A customer is sold a right to logistic service, e.g. represented by a ticket, like a poststamp.

The 'Perform logistic service' transaction is a process that is made up of the subprocesses 'Collect', 'Sort' and 'Distribute'. The goods that have to be transported are first collected from the sender. Next they are sorted according to the location to which they have to be transported, and finally they are distributed to the receiver. There are two imputations, cars and personnel. The logistic service uses both. The sales transaction uses only personnel. Figure 12 shows the subprocesses. The imputations are shown in text for clarity reasons.

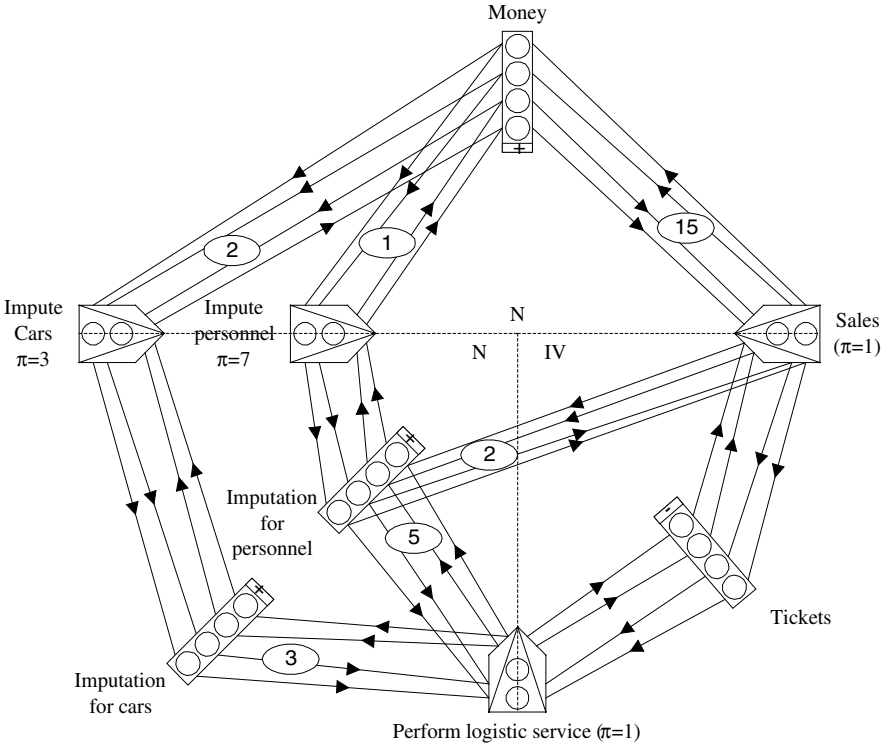


Fig. 11. Logistic enterprise

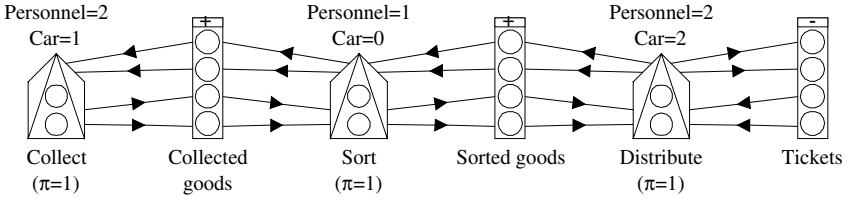


Fig. 12. Perform service subprocesses

The refinement of the perform logistic service process into three subprocesses can be extended to the flow matrix. Consider the sets of buffers and transactions based on the subprocesses: *B'*={'Money', 'Tickets', 'Collected goods', 'Sorted goods', 'Imputation for personnel', 'Imputation for cars'} and *T'*={'Sales', 'Collect', 'Sort', 'Distribute', 'Impute personnel', 'Impute cars'}. The matrix equation that is derived from the structural value cycle is as follows:

$$\begin{bmatrix} 15 & 0 & 0 & 0 & -2 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & -2 & 1 & 0 \\ -2 & -2 & -1 & -2 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 3 \\ 7 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \equiv \begin{bmatrix} J \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \textbf{(2)}$$

Note that the resulting state vector is indeed a vector with zero for all buffers except for the money buffer (see paragraph 3.3). The value 2 that corresponds to the money buffer is the jump value. Since there are no purchase transactions the formula for this jump value reduces to:

$$J = P_{Sales} - \sum_{t \in T' : Imp(t)} \pi(t) Imp_t = 15 - 2 \cdot 3 - 1 \cdot 7 = 2 \quad \textbf{(3)}$$

For the Activity Based Costing and Budgeting system a much more detail was needed. The division of processes into subprocesses was repeated until the required level of detail was reached. Also the proration values and the cardinalities are not constant. The cost of a unit of transport for example depends on the load of a car which in turn depends (among other things) on the total number of products transported. In the system the end-user can specify the necessary functions for computing the variable proration values and cardinalities. The system is used since 1998 by the headquarters of a multinational company.

6. Concluding Remarks and Future Research

The structural value cycle is a normative model of an enterprise. It describes the desired or intended behaviour of the enterprise. The jump value is for example always positive, values are never lost and registration is perfect. Normative models support analysis on the relation between activities to be performed and results obtained. This can be applied for example to Activity Based Costing, Workflow Management Systems, as a norm in the Planning & Control cyclus (cybernetic business control paradigm), Key Performance Indicators and Value Based Management.

The case study was kept rather superficial for reasons of secrecy, since we were involved in a strategic, vital project for the Board of Governors of a large multinational company. Currently we are asked by other companies to develop similar strategical, computerized studies of future business scenario's. As a spin-off of our research we developed a software workbench for value cycles, whose application we plan to intensify in future projects.

References

1. J.H.Blokdijk, F.Drieënhuizen and Ph. Wallage: Reflections on Auditing Theory, A contribution from the Netherlands; Deventer: Kluwer Bedrijfswetenschappen and Amsterdam: Limperg Instituut; 1995.
2. Ph. I. Elsas: Computational Auditing; Ph.D. thesis, Vrije Universiteit, Amsterdam; 1996
3. Ph. I. Elsas and P.M. Ott de Vries: Computing Conspiracies; DEXA 1998
4. Ph. I. Elsas, R.P. van de Riet and J.J. van Leeuwen: Knowledge-Based Audit Support; DEXA 1992
5. A.B. Frielink and H.J de Heer (Eds): Leerboek Accountantscontrole, deel 1, Algemene grondslagen, delen 2a, 2b, De algemene controle, typologie accountantscontrole in het kader van algemene control; Leiden, Antwerpen: Stenfert Kroese; 1985 (Vol .1), 1987 (Vol. 2a) and 1989 (Vol. 2b) (all in dutch)
6. Donald H. Liles and Adrien R. Presley: Enterprise modeling within an enterprise engineering framework; Proceedings of the 1996 Winter Simulation Conference
7. C.A. Petri: Some personal Views of Net Theory; Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II, Lecture Notes in Computer Science, Vol 255
8. E. Smalenbach: Der Kontenrahmen (zweite erweiterte auflage); Fachausschuss für Rechnungswesen; Leipzig: Julius Klinkhardt; 1929 (in German)
9. R.W.Starreveld, H.B.de Mare and E.J.Joëls: Bestuurlijke informatieverzorging, deel 1: Algemene grondslagen, deel 2: Typologie der toepassingen; Alphen aan den Rijn, Brussel: Samson; 1988 (Vol.1) and 1986 (Vol.2) (in Dutch).
10. R.H. Veenstra: Handleiding Assistenten Accountantscontrole; Internal document of Deloitte & Touche: 1972 (in Dutch)

Ontological Design Patterns: Metadata of Molecular Biological Ontologies, Information and Knowledge

Jacqueline Renée Reich

University of Manchester, Department of Computer Science,
Oxford Road, Manchester M13 9PL, United Kingdom
reich@cs.man.ac.uk

Abstract. Within the post-genomic era, scientists want to exploit the information of genome sequencing results, protein interactions, and gene deleted phenotypes. Molecular biologists build ontologies to model their knowledge in non-ambiguous ways. Ontologies are metadata describing the meaning of data. They are useful for query answering in databases, hypotheses testing, and for the representation of context or explanation models. The provision of ontologies of complex information is challenging. Ontological Design Patterns (ODPs) make the ontological structure and content explicit. ODPs can be traced to reuse and adapt ontologies to objectives of information search, analysis and comparison. The three ODPs shown in this paper provide methods to hide unnecessary information (InteractionHider ODP), to notify interdependent concepts if one concept is changed (UpdateDependency ODP), and to handle multiple concepts, that can fulfil a specific context (ChainOfConcept ODP). All ODPs are available from the author with code examples. Contact: reich@cs.man.ac.uk

1 Introduction

The recent availability of complete genome sequences provides biologists with new opportunities for identifying, comparing, and understanding properties of the genome [13]. These new information resources of genomic data sets have heterogeneous or weakly structured formats [19]. To exploit this complex information effective tools for data management, representation, and analysis have to be provided. Consistent and integrated representations of genomic data are a precondition for post-genomic bioinformatics [5]. Therefore, molecular biologists started to build ontologies to model parts of their knowledge and terminology in a concise and non-ambiguous way. Ontologies are a kind of metadata, which describe the meaning or properties of the actual data and provide a consistent data documentation on different levels of abstraction and formalisation. An ontology includes concepts and instances of information, showing their relationships and possible constraints given by the knowledge area. Concepts can have many descriptive attributes, may be partially described at any level of granularity, and may be viewed from many perspectives. A concept summarises features of its instances and names their generalisation. For instance, "flowers" becomes the name of

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 698-709, 2000.

the concept about various kinds of flowers, which is described by attributes (e.g. leaf-colour) common to all its instances. The instances (e.g. the rose on my table) are described by values or objects, and belong to their concept. Concepts or instances can be mutually related (e.g. 'is part of' or 'includes') [15]. The knowledge integrated, stored, and shared within ontologies is useful for query answering in databases, hypotheses testing, and for the representation of informative context or scientific explanation models [3] [10] [12]. The application of ontologies also improves the extraction of scientific text fragments and the evaluation of text contents, because they allow considering the semantics of expressions in different contexts [8].

The construction of an ontology for a molecular biology related theme, such as an intracellular receptor super-family, involves spatial, temporal, and functional limits and choices. For instance, part of an intracellular receptor ontology may define proteins that bind extra-cellular signalling molecules, which then diffuse into the cell across the plasma membrane. The instances of the various receptor and hormone concepts share general features or properties, such as a common origin from cholesterol. This receptor ontology may exclude the related group of cell-surface receptors, because these are located in the plasma membrane with their binding site exposed to the external medium. The decision to integrate or exclude the cell-surface receptors produces a dichotomy to accept or reject part of the general receptor terminology.

Ontologies cannot be part of molecular biology specific software tools if they do not reflect the actual state of the system and knowledge domain. The idea of one global, molecular biological ontology managed by a central repository is unrealistic. The construction process of a global and detailed ontology would be too expensive, and a consistent organisation too difficult caused by the changes in scientific understanding over time. The alternative approach of a federated and distributed management of ontologies presupposes the integration and co-ordination of a kind of meta-meta model [14]. To increase the adaptability of a designed ontology to future demands and to avoid time consuming and expensive redesign, abstractions that emerge during design must be incorporated into the ontology. At this level of abstraction in ontology design, and also at lower levels of its specification and implementation, Ontological Design Patterns (ODPs) are useful [15] [16] [17] [18]. ODPs are a formalisation to ensure that an ontology can be changed and reused in specific ways [4]. Within this paper, three ODPs (i.e. UpdateDependency ODP, InteractionHider ODP, and ChainOfConcept ODP) are applied on the context of intracellular receptors. They will show how ODPs can abstract and identify the design structures and semantic contexts of molecular biological phenomena in an incremental and iterative process, and illustrate the idea of developing and reusing design patterns to solve recurring ontology design problems. The advantages of ODPs are that they 1) make it easier to reuse successful ontology designs and architectures, 2) are successful techniques for developers of new ontologies, 3) help to make the best choice among design alternatives, and 4) improve the documentation and maintenance of existing ontologies by explicitly specifying the interactions between concepts and instances.

The formalisation of ODPs ensures that an ontology can be built in a repetitive fountain-like approach including cycles of specification, classification and implementation. For instance, during a later design cycle, the original fragment of an intracellu-

lar receptor ontology could be definitely modified and linked with a cell-surface receptor ontology changing the context of the original ontology. The capability to adjust and to improve the properties, elements, behaviour, structures, and tasks of ontologies is a precondition that they can follow the development and changes in on-going research and science. Temporal objectives can influence 1) how tightly sub-ontologies or individual concepts are designed and at which degree of granularity, 2) which concept-interfaces and inheritance-hierarchies are defined, and 3) which key-relationships are established among them. For instance, the context of knowledge evaluation may move from a first goal to analyse scientific texts concentrating on intracellular receptor related literature to the goal expanding the analysis to cell-surface receptor related articles. To provide the necessary flexibility to do so, the decomposition, organisation and classification of a scientific vocabulary have to consider the encapsulation, granularity, and dependency of concepts, and the performance, evolution, and reusability of the complete ontology.

So far, ten different ODPs are defined [15] [16] [17] [18], which can be implemented in various knowledge specification or programming languages.

2 Why Patterns?

The idea of ODPs has its roots in a design movement of contemporary architecture [2], literate programming, and the problem-solving discipline of software engineering [9]. The goal of ODPs is to help ontology developers resolve recurring problems of ontology development. ODPs can create a shared language for communicating insight and experience about design problems and their solutions. Formally codifying these solutions and their relationships will capture the body of knowledge, which defines an understanding of good ontology architectures. ODPs can be based upon metaphors of a restricted application domain, they can complement, or elaborate upon conceptual patterns, or they can descend further into implementation details using an ontology specific implementation language. It is the combined presence of all these ODP characteristics that distinguish ODPs from heuristics, rules, or algorithms. Heuristics and principles frequently participate in the concerns of a pattern, but they are only one element of the ODP. A rule may be part of the solution in a pattern description, but a rule solution is neither sufficient nor necessary. ODPs are not designed to be executed or analysed by computers, as one might imagine to be true for rules: ODPs are to be executed with insight, taste, experience, and a sense of aesthetics, as it was described by Coplien for software design patterns [7].

In general, a pattern is an abstraction from a concrete form that conveys the essence of a proven solution to a problem recurring within specific non-arbitrary contexts amidst competing concerns [6] [7]. Each ODP has three parts, which express a relation between a certain context, a problem, and a solution. For example, the concrete form of an `InteractionHider` is the solution to the recurring problem of too numerous interconnections between concepts. The problem occurs within the context of interacting general concepts, and in the presence of the competing concerns, such as con-

ceptual semantic changes within composed, more specific concepts. The proposed solution involves the distinctive structure of `Collaborators` and `InteractionHiders`, which balances these concerns of composed concepts, in the manner most appropriate for the context of a molecular biological receptor ontology. Using the InteractionHider ODP form, the description of the solution, which simplifies concepts and their interfaces, tries to capture and to embody the essential insight of centralising control. This solution should be usable and useful for other ontology designers in similar situations. The ODP also has a name, such as 'InteractionHider ODP', which serves as a conceptual handle, to facilitate discussing the ODP and the information it represents. The successive application of several ODPs, each encapsulating its own problem and concern, such as the further ChainOfConcept ODP and the UpdateDependency ODP, unfolds a larger solution, which emerges indirectly as a result of the smaller solutions.

Writing good ODPs is very difficult, because an ODP should help its users to 1) comprehend existing systems, 2) customise systems to fit user needs, and 3) construct new systems. This will be illustrated by the following three examples of an InteractionHider ODP, an UpdateDependency ODP, and a ChainOfConcepts ODP, which are implemented in the programming language Common Lisp Object System (CLOS) [20]. ODP examples are applied within the case study of molecular biological receptors, which distinguish two groups of cell-surface receptors and intra-cellular receptor. A receptor is a protein that binds a specific extra-cellular signalling molecule, called a ligand, and initiates a response in the cell. Cell-surface receptors, such as the acetylcholine receptor and the insulin receptor, are located in the plasma membrane, with their ligand-binding site exposed to the external medium. Intra-cellular receptors, such as steroid hormone receptors, bind ligands, that diffuse into the cell across the plasma membrane. A steroid is a hydrophobic molecule related to cholesterol. Many important hormones, such as estrogen and testosterone, are steroids. The intracellular receptor super-family includes thyroid hormone receptors, cortisol receptors, estrogen receptors, progesterone receptors, Vitamin D receptors, and retinoic acid receptors [1].

3 Interaction-Hider ODP

Ontology design encourages the organisation of specific concepts into more general concepts and the distribution of their meaning. More specific concepts, such as 'cell-surface receptor' or 'intra-cellular receptor', are often larger expressions composed of several words; more general concepts, such as 'receptor', are often shorter expressions and sometimes described by a single word. The distribution of concepts, which form and compose more specific concepts, can result in a deep concept structure with many connections between the conceptual components. In the worst case, each concept ends up knowing about all others. Though partitioning an ontology into many higher and lower concepts generally enhances reusability, proliferating interconnections tend to reduce it again. Numerous interconnections make it less likely that a combination of

concepts is understandable, recognisable and correct, and the ontology will become monolithic.

A possible solution is to encapsulate the meaning of a more complex and specific concept, such as 'steroid hormone receptor', by a separate `InteractionHider` concept. An `InteractionHider` is responsible for controlling and co-ordinating the interactions of higher and more general concepts, such as the concepts of 'steroid', 'hormone' and 'receptor'. The `InteractionHider` serves as an intermediary step, that keeps concepts grouped into more specific related sub-concepts from referring to each other, and explicitly promotes loose connecting. The individual concepts of 'steroid', 'hormone' and 'receptor' only know the `InteractionHider`, thereby reducing the number of interconnections. Therefore, the InteractionHider ODP defines how to encapsulate 1) the interaction of more general concepts, 2) the relationship of their meanings, and 3) the correct semantic changes of the individual concepts when they get composed with each other.

The InteractionHider ODP improves the reuse of heavily inter-linked general concepts. Meaning that is distributed between several concepts will be customisable, because the `InteractionHider` selects, analyses and adapts the meaning and information distributed among several general concepts. To change any meaning requires the specification of various concrete `InteractionHiders`, such as 'steroid-hormone-receptor'. The loosely connected `Collaborator` concepts, which are the components of the more specific concepts, can be used as they are. The distinction of `Collaborator` and `InteractionHider` allows to vary and to reuse them independently, to simplify concepts and their interfaces, and to centralise control. The `InteractionHider` and its subclasses define an interface to compose concepts into more specific concepts. The interface communicates with `Collaborators`, which access the general concepts. The actual `InteractionHider` 'steroid-hormone-receptor' knows, maintains, and co-ordinates the `Collaborators` 'steroid', 'hormone', and 'receptor'. Each `Collaborator` knows its `InteractionHider`. Each `Collaborator` communicates with its `InteractionHider` whenever it would have otherwise interacted with another `Collaborator` directly.

I am using a `ReceptorConceptComposer`, which is a kind of `InteractionHider`, to implement the above receptor example. The class `ReceptorConceptComposer` defines the interface for composers of receptor related concepts. The interface has the function of creating and composing conceptual elements.

```
(defclass InteractionHider ()())
(defclass ReceptorConceptComposer (InteractionHider)())
(defmethod composeConceptEl ((ob ReceptorConceptComposer))())
(defmethod createCollaborator ((ob ReceptorConceptComposer))())
```

Subclasses of `ReceptorConceptComposer`, such as `Organiser`, will override `composeConceptEl` to affect the appropriate concept element. These subclasses will also override `createCollaborator` to construct the concept elements in the more specific receptor concepts.

`Collaborator` is the class for concept elements, which are referenced to compose more specific concepts. In this example, each concept element knows its `conceptComposer`.

```
(defclass Collaborator ()
   ((conceptComposer :initarg :conceptComposer)))
(defmethod get-conceptComposer ((ob Collaborator))
   ((slot-value ob 'conceptComposer)))
```

Concept elements call the composeConceptEl operation on their concept composers. In this example, composeConceptEl of Collaborator calls the operation composeConceptEl of the concept composer for receptors.

```
(defmethod composeConceptEl ((ob Collaborator))
   (composeConceptEl (get-conceptComposer ob )))
```

Steroid, Hormone, and Receptor are subclasses of Collaborator. They all provide operations to combine, select, count, and to arrange the components for more specific concepts, such as "steroid hormone", "steroid hormone receptor", "thyroid hormone receptor", and "retinoic acid receptor". All these receptor examples are structurally related in a molecular biological sense and belong to the same intracellular receptor super-family. Only the code for Receptor is shown; the code for Steroid and Hormone are analogous.

```
(defclass Receptor (Collaborator)())
(defmethod combine ((ob Receptor) els)
   ((composeConceptEl ob)))
(defmethod arrange ((ob Receptor) els)
   ((composeConceptEl ob)))
(defmethod select ((ob Receptor) els)())
(defmethod count ((ob Receptor) concept)())
```

The Organiser class mediates between the components of the various receptor concepts. Organiser is a subclass of ReceptorConceptComposer. Organiser keeps track of the components, which it arranged within the concept. It redefines the createCollaborator to create the concept components and to initialise its references to them. The inherited operation composeConceptEl has to ensure that the concept components collaborate properly. It becomes obvious that the complexity of composeConceptEl increases proportionally with the complexity of the concepts related to various receptors.

```
(defclass Organiser (ReceptorConceptComposer)())
(defmethod createCollaborator ((ob Organiser))
   ("get all concept elements"
   "assemble and arrange the components in the concept"))
```

4 Update-Dependency ODP

A common side effect of partitioning an ontology into a collection of collaborating concepts is the need to maintain consistency between related concepts. This necessity was already quite obvious in the previously discussed InteractionHider ODP when composing general concepts into more specific concepts of the intracellular receptor

super-family. This super-family includes thyroid hormone receptors, cortisol receptors, estrogen receptors, progesterone receptors, Vitamin D receptors, and retinoic acid receptors [1]. Consistency within the concepts of the various receptors should not be achieved by making the concepts and their sub-components, such as Vitamin D or estrogen, tightly connected, because that will reduce their reusability. It should be possible to adapt an individual concept and all related information to any kind of change also if it is unknown how many concepts need to be changed finally. For instance, the ‘intracellular receptor super-family’ can also be called the ‘steroid-hormone receptor super-family’. If such a change of naming would occur within an ontology with tightly coupled concepts then concepts like 'steroid' and 'hormone' would be directly affected. Therefore, a concept with any degree of specificity should be able to notify other concepts without making assumptions about which these concepts are.

The UpdateDependency ODP defines a one-to-many dependency between concepts so that when one concept changes significantly, all its dependents are notified and updated automatically. The key elements in this ODP are `Elements` and `DependencyUpdaters`, between which only an abstract and minimal connection exists.

An `Element` maintains and knows a list of its `DependenyUpdaters`, each conforming to the simple interface of the abstract `DependencyUpdater` concept. It also provides an interface for attaching and detaching `DependencyUpdaters`. Any number of `DependencyUpdaters` may update an `Element`. The updated element does not know which `DependencyUpdater` actually changed it. All `DependencyUpdaters` are notified whenever the `Element` undergoes a change.
A `DependencyUpdater` defines an updating interface for concepts that should be notified of changes in an `Element`. The `DependencyUpdater` that initiates the change request postpones its update until it gets a notification from the `Element`. An unexpected change of an `Element` may cause a cascade of updates to `DependencyUpdaters` and their dependent concepts, because `DependencyUpdaters` have no knowledge of each other's presence. Therefore, a protocol about the dependency criteria has to be well-defined and maintained that they can be tracked down and their cost evaluated.

For example, the change of the concept 'intracellular receptor super-family' into 'steroid-hormone receptor super-family' would require the update of all depending information, such as the definition of the composed but still general concept 'steroid hormone receptor'. Steroid hormones include cortisol, the steroid sex hormones, vitamin D in vertebrates, and the moulting hormone ecdysone in insects. All these hormones are made from cholesterol, which could also form a separate concept and a set of direct links to the various hormones further expanding to the appropriate receptors.
The `Element` 'steroid hormone' stores interesting information for any `DependencyUpdater`, such as 'receptor super-family'. The `Element` 'steroid hormone' sends notification to its `DependencyUpdaters` when some information changes. The `DependencyUpdater` 'receptor super-family' also maintains a reference to the `Element` 'steroid hormone'. The `DependencyUpdater` 'receptor super-family' implements the interface to store information that should stay consistent with the up-

dated Element, which would be in this example the naming change from 'intracellular receptor super-family' into 'steroid-hormone receptor super-family'.

An abstract concept defines the DependencyUpdater interface and supports multiple Elements for each DependencyUpdater. The Element passed to the update operation of DependencyUpater lets the DependencyUpdater determine which Element changed when it knows about a list of potential elements to be updated.

```
(defclass DependencyUpdater ()())
(defmethod update ((ob DependencyUpdater)Element)())
```

Similar to DependencyUpdater, an abstract concept defines the Element interface showing the functions to add and remove potential DependencyUpdaters related to the conceptual Element:

```
(defclass Element ()
   ((dependencyUpdaters :initarg :dependencyUpdaters)))
(defmethod attach ((ob Element) dependencyUpdater)
((setf (slot-value ob 'dependencyUpdaters)
(append dependencyUpdater (slot-value ob 'dependencyUpdaters))))
(defmethod detach ((ob Element) dependencyUpdater)
((setf (slot-value ob 'dependencyUpdaters)
(remove dependencyUpdater (slot-value ob 'dependencyUpdaters))))
(defmethod notify ((ob Element))
   (dolist (i (slot-value ob 'dependencyUpdaters))
           (update i currentItem))
```

SteroidHormone is an Element involved in naming 'intracellular receptor super-family' synonymously as 'steroid hormone receptor super-family'. SteroidHormone notifies its DependencyUpdaters about any semantic change caused by added suffixes or adverbs, or new word compositions. SteroidHormone also provides the interface for retrieving and accessing individual information about cortisol, the steroid sex hormones, vitamin D, and the hormone ecdysone.

```
(defclass SteroidHormone (Element)())
(defmethod get-cortisol ((ob SteroidHormone))())
(defmethod get-sex-hormones ((ob SteroidHormone))())
(defmethod get-vitamin-D ((ob SteroidHormone))())
(defmethod get-ecdysone ((ob SteroidHormone))())
```

The change operation is called to provide an accurate basis of word and expression components. change updates the SteroidHormone's internal state and calls notify to inform DependencyUpdaters of the change:

```
(defmethod change ((ob SteroidHormone))
   (;update internal state of the expression components
   (notify ob))
```

5 Chain-of-Concepts ODP

When an original context gets initiated within an ontology, the concepts that ultimately will fulfil the meaning and represent the information might not be known explicitly. For example, the context might be that steroid hormones are all made from cholesterol, and that all act by a similar mechanism diffusing directly across the plasma membrane of target cells and binding to intracellular receptor proteins [1]. We can imagine the situation, when the potential concepts of cortisol, steroid sex hormones, vitamin D, and the moulting hormone ecdysone hormones fulfilling the context of steroid hormones were not known from beginning but specified dynamically. The ChainOfConcepts ODP uncouples contexts and concepts by giving multiple concepts a chance to fulfil the context. The receiving concepts are forming a chain, and the context is passed along this chain until it gets fulfilled. In our case study, the concepts of cortisol, steroid sex hormones, vitamin D, and ecdysone would form such a chain. To forward the context along the chain and to ensure that the receiving concepts remain unknown, each concept on the chain shares a common interface for expressing a context and for accessing its successor on the chain. This chaining technique simplifies concept interconnections. Instead of concepts maintaining references to all candidate concepts, they keep a single reference to their successor. The information and meaning expressed by a context will be modified through changes in the chain of concepts. Since a context has no explicitly related concept there is no guarantee that it will be finally expressed, especially if the chain of concepts is not configured properly.

The `PotentialConcept` defines an interface for handling and expressing contexts, and it implements the successor link. Within the chain, each concrete concept, such as `Vitamin-D`, `steroid sex hormone`, `cortisol`, and `ecdysone`, accesses its successor and forwards the context, if it cannot express the context itself.

```
(defclass PotentialConcept ()
   ((successor   :initarg :successor)
    (context     :initarg :context)))
(defmethod get-successor ((ob PotentialConcept))())
(defmethod handle-context ((ob PotentialConcept))
   ((if (get-successor ob)
        (handle-context (get-successor ob)))))
(defmethod fulfil-context ((ob PotentialConcept))())
```

All concrete concepts are subclasses of the abstract `PotentialConcept` class.

```
(defclass ConcreteConcept (PotentialConcept)())
```

In this example, Vitamin D is the first concrete concept on the chain. `Vitamin-D` is a subclass of `ConcreteConcept`.

```
(defclass Vitamin-D (ConcreteConcept)())
```

The `handle-context` operation of `Vitamin-D` first tests if there is a context it can fulfil. If the ontology designer has defined none, then the context gets for-

warded to the successor using the `handle-context` operation of `Potential-Concept`.

```
(defmethod handle-context ((ob Vitamin-D))
   ((if (fulfil-context ob)
        ;express and fulfil the context. If not then...
        (call-next-method ob))))
```

The rest of the concrete concepts will implement a similar scheme, such as:

```
(defclass SteroidSexHormone (ConcreteConcept)())
(defclass Cortisol (ConcreteConcept)())
(defclass Ecdysone (ConcreteConcept)())
```

At the end of the chain is a concept called `steroidHormone`. `SteroidHormone` is not a concrete concept, but a direct subclass of `PotentialConcept`. Whenever a `fulfil-concept` operation propagates to this level of generality, because no more specific concept could fulfil the context, the concept `SteroidHormone` can supply information in general, or it can offer a list of different contexts.

```
(defclass SteroidHormone (PotentialConcept)())
(defmethod fulfil-context ((ob SteroidHormone))
   ((;show a list of related or alternative contexts)))
```

The operation `fulfil-context` can be called from any concrete concept in the chain. A called concrete concept may fulfil the context immediately. Any concrete concept can be the successor of any other one, and the successors can be changed dynamically to get the proper context expressed wherever the search is started within the concept chain.

6 Conclusions

Ontological Desing Patterns (ODPs) try to provide a technique for the design and implementation of genome-level ontologies that are important to post-genomic bioinformatics. These ontologies have to be described in a consistent and coherent manner. The objectives of ODPs are 1) to improve the interaction with and within the ontology revealing where the data came from, and what happened to it, 2) to improve data quality, which has to be consistent, up-to-date, accurate and complete, 3) to support any system integration process of individual data sources making available the descriptions and interfaces of various ontologies, and 4) to support the maintenance, analysis, and design of new ontologies by increasing the control and reliability of the ontology development process providing information about the structure, meaning, and origin of the ontological concepts.

ODPs solve design problems and capture solutions, not just abstract principles or strategies. The solutions offered by an ODP are not always quite obvious. ODPs try to

describe relationships, deeper system structures and mechanisms, and not just modules. The best ODPs generate a solution to a problem indirectly, which is a necessary approach for the most difficult problems of design. This paper provides the descriptions and code examples of the three ontological design patterns InteractionHider ODP, UpdateDependency ODP, and ChainOfConcept ODP. As shown by these ODP examples applied on intracellular receptors, ODPs describe the meaning or properties of data with the aim to better understand, manage and use that data. They provide a consistent documentation about the structure, the development process and the use of an ontology, because a specific ODP pattern models information behind the actual data. ODPs are difficult to develop, and the construction of the ODPs presented in this paper was an iterative process. As conceptual representations of ontological design, these ODPs could equally be mapped to different knowledge representation languages, such as Description Logics [5] [11] or XML (http://www.w3.org/TR/WD-xml-link).

Acknowledgements

This work was done at the Department of Computer Science of the University of Manchester. Many thanks to William Bennett for reviewing the paper.

References

1. Alberts, B.; Bray, D.; Lewis, J.; Raff, M.; Toberts, K.; Watson, J. D. (1994). Molecular Biology of the Cell. 3rd ed. Garland Publishing, Inc. New York, London
2. Alexander, Chr. (1979). The Timeless Way of Building, Oxford University Press
3. Bench-Capon, T. (1998). The Role of Ontologies in the Verification and Validation of Knowledge based Systems. Proceedings of the 9th International Workshop on Database and Expert Systems, 64-69. Los Alamitos, IEEE Press,
4. Beys, P.; Benjamins, R.; Van Heijst, G. (1996). Remedying the reusability-usability tradeoff for problem-solving methods. Proceedings of the Tenth Knowledge Acquisition Workshop (KAW96)
5. Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Nardi, D.; Rosati, R. (1998). Description logic framework for information integration. Proceedings of the sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR-98): 2-13
6. Coad, P. (1992). Object-oriented Patterns. Communications of the ACM. 35(9):152-159
7. Coplien, J.O. (1992). Advanced C++ Programming Styles and Idioms. Addison-Wesley, Reading, MA
8. Craven, M.; Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology. Heidelberg, Germany:77-86
9. Gamma E., Helm R., Johnson R., Vlissides J. (1994). Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley
10. Gruber, Th. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. International Journal Human-Computer Studies 43, 907-928

11. Horrocks, I. R. (1998). Using an expressive description logic: FaCT or Fiction? Proceedings of the sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)
12. Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W. (1991). Enabling technology for knowledge sharing. AI Magazine: 36-56
13. Oliver, S. (1996). From DNA sequence to biological function. Nature, 379:597-600.
14. Prekas, N.; Loucopoulos, P.; Roland, C.; Grosz, G.; Semmak, F.; and Brash, D. (1999). Developing Patterns as a Mechanism for Assisting the Management of Knowledge in the Context of Conducting Organisational Change. Proceedings of the 10th International Conference, DEXA'99, Florence Italy, Lecture Notes in Computer Science 1677, Springer Verlag
15. Reich, J.R. (2000 b). Modelling Molecular Biological Information: Ontologies and Ontological Desing Patterns. IRMA 2000, 21-24 May, Anchorage, Alaska 166 (http://www.irma-international.org/call2000.htm)
16. Reich, J.R. (2000 a). Ontological Design Patterns for the Management of Molecular Biological Knowledge. AAAI 2000, 20-22 March, Stanford, CA, USA (http://www.aifb.uni-karlsruhe.de/~sst/Research/Events/sss00/)
17. Reich, J.R. (1999 b). Molecular Biological Ontologies and Ontological Design Patterns: What they are, and how they look like. IEE Informatics, Two-day Seminar 'Searching for information: artificial intelligence and information retrieval approaches', 11.-12.November, Glasgow, Great Britain: 12/1-12/3
18. Reich, J. R. (1999 a). Ontological Design Patterns for the Integration of Molecular Biological Information. Proceedings of the German Conference on Bioinformatics (GCB'99): 156-165, Hannover, Germany, (http://www.bioinfo.de/isb/gcb99/talks/reich)
19. Robbins, R. J. (1996). Comparative Genomics: a new Integrative Biology. In: Julio Collado-Vides, Boris Magasanik, Temple F. Smith (eds.). Integrative Approaches to Molecular Biology, The MIT Press, Cambridge: 63-90
20. Steele, G. L., Jr. (1990). Common Lisp the Language, 2d edition. Digital Press

Exploiting the Duality of Maximal Frequent Itemsets and Minimal Infrequent Itemsets for I/O Efficient Association Rule Mining

K. K. Loo 葉志立 Yip Chi Lap Ben Kao David Cheung

Department of Computer Science and Information Systems,
The University of Hong Kong, Hong Kong.
{kkloo, clyip, kao, dcheung}@csis.hku.hk

Abstract. Any algorithm for mining association rules must discover the set of all maximal frequent itemsets ($maxL$) from a database. Given a set of itemsets X, to verify that X is $maxL$, two conditions must be checked: (1) any itemset x in X is frequent, and (2) the *dual* of X must be the set of all minimal infrequent itemsets ($minS$). This observation leads us to a family of algorithms for mining association rules. Given a reasonable guess of $minS$ and $maxL$, we verify their duality relationship, and refine the two sets until the above two conditions hold. We note that previously proposed algorithms such as `Apriori` and `Pincer-Search` are all members of our algorithm family. Also, we study a member algorithm called `FlipFlop`. Through a series of experiments, we show that `FlipFlop` significantly reduces the I/O requirement of mining association rules.

Keywords: Data mining, association rules, lattice

1 Introduction

Association rule mining [2] is one of the hottest topics in data mining. Recent interesting works on this problem and its variations include [1, 4, 5]. An association rule is an induction rule of the form $X \rightarrow Y$, where X and Y are two non-empty and non-overlapping sets of items. The problem of association rules mining can be decomposed into two subproblems:

1. Find out all *frequent itemsets* and their support counts. A frequent itemset is a set of items which are contained in a sufficiently large number of transactions, with respect to a support threshold *minimum support*.
2. From the set of frequent itemsets found, find all association rules that have a confidence value exceeding a confidence threshold *minimum confidence*.

Since the solution to the second subproblem is straightforward [3], major research efforts have been spent on the first subproblem. Most of the algorithms devised to find the set of all frequent itemsets, L, are based on the `Apriori` algorithm [3]. The `Apriori` algorithm finds out the frequent itemsets iteratively. In the i^{th} iteration, `Apriori` generates a number of candidate itemsets of size

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 710–719, 2000.

i. `Apriori` then scans the database to find the support count of each candidate itemset. Itemsets whose support counts are smaller than the minimum support are discarded. `Apriori` terminates when no more candidate set can be generated.

The key of the `Apriori` algorithm is the `Apriori_Gen` function which wisely generates only those candidate itemsets that have the potential of being frequent. However, at each database scan, only candidate itemsets of the same size are generated. Consequently, the number of database scans required by `Apriori`-based algorithms depends on the size of the largest frequent itemsets. As an example, if a database contains a size-10 frequent itemset, then at least 10 database scans are required. For large databases containing gigabytes of transactions, the I/O cost is dauntingly big. The goal of this paper is to devise an I/O efficient algorithm for finding all frequent itemsets. As we will see soon, our trick lies on the duality property of *maximal frequent itemsets* and *minimal infrequent itemsets*.

Among all the frequent itemsets in L, some of them are *maximal*, that is, they are not proper subsets of any other frequent itemset in L. If one is given the set of all these *maximal* frequent itemsets (denoted by $maxL$), then finding the support counts of all frequent itemsets would become very simple. Essentially, one has to (1) generate, for each maximal frequent itemset $y \in maxL$, its down-set $\downarrow y = \{x | x \subseteq y\}$ (i.e., the set of all subsets of y); (2) union all these down-sets to obtain $\bigcup_{y \in maxL} \downarrow y$; (3) scan the database once to obtain the support count of every itemset in $\bigcup_{y \in maxL} \downarrow y$. Notice that by the property that an itemset is frequent implies all its subsets are frequent, we have $\bigcup_{y \in maxL} \downarrow y = L$. So, the above steps would obtain the support counts of all frequent itemsets. Our discussion thus suggests that if one could find $maxL$ efficiently, then one could potentially reduce the number of database scans required for the mining process.

Before we discuss how to obtain $maxL$ fast, let us first consider the *dual* of the problem. Suppose U is the set of all itemsets, then the set of all infrequent itemsets, S, would be equal to $U - L$. Among all the itemsets in S, some of them are *minimal* in the sense that they are not proper supersets of any other itemset in S. Let us denote the set containing all these minimal infrequent itemsets by $minS$. Now, given $maxL$, one can (theoretically) *uniquely* determine $minS$ by evaluating the following expression:

$$minS = \top(maxL) = \min(U - (\bigcup_{y \in maxL} \downarrow y)),$$

where min() is a function which, given a set Y, returns all minimal itemsets in Y We use the symbol $\top()$ to denote the transformation from a $maxL$ to its corresponding $minS$. Conversely, we note that given $minS$, $maxL$ can be determined by:

$$maxL = \bot(minS) = \max(U - (\bigcup_{y \in minS} \uparrow y)),$$

where $\uparrow y = \{x | x \in U \wedge y \subseteq x\}$ (called the up-set of y) is the set of all supersets of y, and max() returns all the maximal elements of a given set. We use the symbol $\bot()$ to denote the transformation from a $minS$ to its corresponding $maxL$. Given

1 Algorithm `Find-Freq-Item-Set`($\widehat{minS}$, $\widehat{maxL}$)
2 while(some itemset x in $\widehat{minS}$ is frequent OR
some itemset y in $\widehat{maxL}$ is infrequent OR
$\widehat{minS} \neq \top(\widehat{maxL})$ OR $\widehat{maxS} \neq \bot(\widehat{minS})$)
3 refine $\widehat{minS}$
4 refine $\widehat{maxL}$
5 end while
6 $L = \bigcup_{z \in \widehat{maxL}} \downarrow z$
7 scan the databse and count the supports of the itemsets in L whose supports have not been counted yet.

Fig. 1. Finding the support counts of all frequent itemsets

a set of maximal itemsets Y and a set of minimal itemsets X, if $X = \top(Y)$ (or equivalently, $Y = \bot(X)$), we say that X and Y are duals of each other.

Suppose an oracle suggests that $\widehat{minS}$ and $\widehat{maxL}$ are his *guesses* of the real $minS$ and $maxL$, respectively.[1] To verify that they are the true $minS$ and $maxL$, they have to satisfy the following necessary and sufficient conditions:

(1) $\forall x \in \widehat{minS}, x$ is infrequent;
(2) $\forall x \in \widehat{maxL}, x$ is frequent;
(3) $\widehat{minS} = \top(\widehat{maxL})$ (or equivalently, $\widehat{maxL} = \bot(\widehat{minS})$).

Proof: See [7].

If all three conditions hold, then we have $\widehat{maxL} = maxL$ and the task of finding the support counts of all frequent itemsets can be done easily as discussed previously. However, if some of the conditions do not hold (e.g., if some itemset x in $\widehat{minS}$ is actually frequent), we need to *refine* the sets $\widehat{minS}$ and $\widehat{maxL}$. The refined sets can be tested against the three conditions again, and be refined further again if not all of the conditions are satisfied. This strategy leads us to a framework of algorithms for finding frequent itemsets as shown in Figure 1.

We show in [7] that previous proposed algorithms, e.g. `Apriori`, are special cases of our framework. In the rest of the paper, we address the following issues:

- How to check the conditions? Checking conditions (1) and (2) above can be easily done by scanning the database. However, checking the duality condition may not be straightforward. We will show how the dual operators can be implemented efficiently in Section 2.
- How to obtain an initial "good" guess of $minS$ and $maxL$? In Section 4, we will show how to obtain a good guess of $minS$ and $maxL$ by mining a small sample of the database.
- How would the algorithm framework lead to an efficient algorithm? We propose and discuss an algorithm `FlipFlop` in Section 3, which takes advantage of an educated guess of $maxL$ and $minS$ to discover all frequent itemsets.

[1] We assume that $\widehat{minS}$ and $\widehat{maxL}$ contain only minimal and maximal itemsets, respectively. That is, $\widehat{minS} = \min(\widehat{minS})$ and $\widehat{maxL} = \max(\widehat{maxL})$.

2 The dual operators

2.1 Computing $\top(Y)$

Let us first consider $\top()$. A brute force algorithm would enumerate all itemsets in U, except those that are subsets of any itemset in Y. The resulting set A is $U - \bigcup_{y \in Y} \downarrow y$. The algorithm then computes $\min(A)$ by discarding all non-minimal itemsets in A. This method is clearly infeasible, since the cardinality of U is exponential to the number of items in the database. However, the following observations tell us that we do not need to consider all the itemsets in U.

Observation 1 *If k is the size of the largest itemset in Y, then no itemset x in $\top(Y)$ contains more than $k+1$ items.*

Observation 2 *For any itemset $x = \{x_1, ..., x_n\} \in \top(Y)$, we must have $\forall i$ in $[1, n], \exists t_i \in Y$ such that $x - \{x_i\} \subseteq t_i$. Moreover, if $n \geq 2$, then $\nexists t \in Y$ such that for two distinct x_i and x_j in x, both $x - \{x_i\} \subseteq t$ and $x - \{x_j\} \subseteq t$.*

Observation 1 suggests that in order to compute $\top(Y)$, one only needs to consider itemsets of size at most $k+1$, where k is the size of the largest itemset in Y. Also, for any itemset $x = \{x_1, ..., x_n\}$ $(2 \leq n \leq k+1)$, if $x \in \top(Y)$, Observation 2 implies that there must exist two distinct $t_1 \in Y$ and $t_2 \in Y$ such that $x - \{x_1\} \subseteq t_1$ and $x - \{x_2\} \subseteq t_2$. The intersection of t_1 and t_2 must therefore contain at least $n-2$ items (in particular, $\{x_3, ..., x_n\}$). Also, note that $x_1 \in t_2 - t_1$ and $x_2 \in t_1 - t_2$. For proofs of the observations, please see [7].

Based on the observations, our strategy of computing $\top(Y)$ goes as follow: (1) Determine k, the size of the largest itemset in Y. (2) For each $2 \leq n \leq k+1$, consider every pair of t_i, t_j in Y such that $|t_i \cap t_j| \geq n-2$, then generate all itemsets x by picking $n-2$ items from $t_i \cap t_j$, one item from $t_i - t_j$, and one item from $t_j - t_i$, and verify that every size-$(n-1)$ subset of x is a subset of some $t \in Y$, and that x itself is not a subset of some $t \in Y$. (3) Finally, we add all size-1 itemsets which are not subsets of any $y \in Y$. Figure 2 shows the algorithm for computing $\top_n(Y)$, which, given a maximal set Y, returns all itemsets in $\top(Y)$ whose size equals $n \geq 2$. Figure 3 shows the algorithm for computing $\top(Y)$.

2.2 Computing $\bot(X)$

In `Pincer-Search` [6], the authors invented an ingenious way of discovering frequent itemsets. Informally, `Pincer-Search` always maintains a set called *MFCS* which satisfies the constraint that any frequent itemset in the database must be a subset of some itemset in *MFCS*. By scanning the database, `Pincer-Search` may discover that a certain set of itemsets, A, are infrequent. A function called `MFCS-gen` is then executed to refine *MFCS* so that the newly discovered knowledge (due to A) is reflected in the new *MFCS*. Due to space limitation, interested readers are referred to [6] for details.

We found that `MFCS-gen` is indeed an efficient way of computing $\bot$ in disguise. Figure 4 shows an algorithm for computing $\bot(X)$ given a set of minimal itemsets X. We remark that our algorithm originates from `MFCS-gen` proposed in [6].

1 function $\top_n(Y)$
2 $X := \emptyset$
3 foreach $t_i, t_j \in Y$, $t_i \neq t_j$,
4 if $(|(t_i \cap t_j)| \geq n-2)$
5 $$\Delta := \left\{ \{a_1, a_2, ..., a_n\} \;\middle|\; \begin{array}{l} a_1, a_2, ..., a_{n-2} \in t_i \cap t_j, \\ a_{n-1} \in t_i - t_j,\ a_n \in t_j - t_i, \\ \forall d \in [1, n]\ \exists t \in Y \text{ s.t.} \\ \quad \{a_1, a_2, ..., a_n\} - \{a_d\} \subseteq t, \\ \nexists y \in Y \text{ s.t. } \{a_1, a_2, ..., a_n\} \subseteq y \end{array} \right\}$$
6 $X := X \cup \Delta$
7 return X

Fig. 2. Computing $\top_n(Y)$

1 function $\top(Y)$
2 $k :=$ size of the largest itemset in Y
3 $X := \emptyset$
4 for $p := 2$ to $k+1$
5 $X := X \cup \top_p(Y)$
6 $X = X \cup \{\{a\} | a \notin y\ \forall y \in Y\}$
7 return X

Fig. 3. Computing $\top(Y)$

The $\perp()$ algorithm maintains a set of itemsets Y, which, when the algorithm terminates, gives $Y = \perp(X)$. Initially, Y is set to contain the lone set of all items in the database. $\perp()$ then processes each itemset s in X in turn. If any itemset m in Y is a superset of s, we know that m must not in $U - \bigcup_{x \in X} \uparrow x$, and hence, $m \notin \perp(X)$, and should thus be removed from Y (line 7). We then generate all the *immediate* subsets of m that can be formed by removing some item $e \in s$ from m, since these itemsets could still be in $U - \bigcup_{x \in X} \uparrow x$. We then check whether they are maximal in Y. If so, they are added to Y.

3 FlipFlop

One nice property of `Apriori` is that with $\widehat{maxL} = max(\bigcup_{j=1}^{i-1} L_j)$ as its guess of $maxL$, verifying that the dual of $\widehat{maxL}$ contains only infrequent itemsets amounts to only checking the itemsets in C_i. The verification process is thus very simple. Unfortunately, the special structure of `Apriori`'s $\widehat{maxL}$ essentially requires that the complete set of all L_j $(j < i)$ be known in order to derive what size-i itemsets (i.e., the set C_i) need to be checked. Also, the result of the checking would supply `Apriori` with the information of L_i only, but not the itemsets of size larger than i. This implies that the refinement of $\widehat{maxL}$ must be performed in steps, with L_1 being `Apriori`'s initial guess of $maxL$. This guess, however, is usually "fairly far" from the real $maxL$. It thus seems that the first database scan is not effectively utilized to obtain a good guess.

```
1   function ⊥(X)
2       Y := {u}
3       for each itemset s ∈ X
4           for each itemset m ∈ Y
5               if s ⊆ m
6                   Y := Y − {m}
7                   for each item e ∈ s
8                       if m − {e} is not a subset of any itemset in Y
9                           Y := Y ∪ {m − {e}}
10      return Y
```

Fig. 4. Computing $\perp(X)$

```
1   algorithm FlipFlop(m̂axL)
2       CS = (⋃_{x∈m̂axL} ↓ x) ∪ ⊤(m̂axL)
3       m̂axL := m̂inS := ∅
4       repeat
5           scan database and count the supports of all itemsets in CS
6           FCS := all frequent itemsets in CS
7           IFCS := all infrequent itemsets in CS
8           m̂axL := max(m̂axL ∪ FCS)
9           m̂inS := min(m̂inS ∪ IFCS)
10          CS := (⊤(m̂axL) − m̂inS) ∪ (⊥(m̂inS) − m̂axL)
11      until CS = ∅
12      return the support counts of all itemsets in ⋃_{x∈maxL} ↓ x
```

Fig. 5. Algorithm `FlipFlop`

Alternatively, one can take a sample of the database and obtain a more reasonable guess of $maxL$ by mining the sample. In a previous study [9], it is shown that sampling is a cost-effective technique for finding an approximate solution to the mining problem. As an example, if the size of the largest frequent itemset is 10, then mining a 10% sample using `Apriori` requires scanning the 10% sample about 10 times. The I/O cost is about the same as that of scanning the complete database once to obtain L_1. However, the $\widehat{maxL}$ obtained by the mine-the-sample method would be much closer to the real $maxL$ than L_1.

A more efficient mining algorithm should thus make use of a better guess of $maxL$. We propose an algorithm `FlipFlop` that follows this principal. Given a $\widehat{maxL}$ (perhaps obtained by mining a sample of the database), `FlipFlop` first computes the dual $\widehat{minS} = \top(\widehat{maxL})$. All itemsets in $\widehat{minS}$ and $\bigcup_{x \in \widehat{maxL}} \downarrow x$ are put into a counting set, CS. The database is then scanned to obtain the support counts of all the itemsets in CS. (Any itemset that is a subset of some itemset in $\widehat{maxL}$ is included in CS because if $\widehat{maxL}$ is a reasonable guess of $maxL$, then most of the itemsets in $\bigcup_{x \in \widehat{maxL}} \downarrow x$ are frequent and their supports are needed anyway to generate all association rules. Thus we might count them

Parameter	Description	Value
$\|D\|$	database size in number of transactions	131,072
N	number of items	1,000
$\|W\|$	number of potentially frequent itemsets	2,000
$\|I\|$	average size of potentially frequent itemsets	4
$\|T\|$	average size of transactions	20
S_q	group size - 1	6

Table 1. Parameters of the database generation model

as well in the first pass.) The set CS is then partitioned into two sets: the set of frequent itemsets, FCS, and the set of infrequent itemsets, $IFCS$. The result of the counting is used to update $\widehat{minS}$ and $\widehat{maxL}$. Specifically, $\widehat{maxL} = \max(FCS)$ and $\widehat{minS} = \min(IFCS)$. At this point, we know that $\widehat{maxL}$ contains only maximal frequent itemsets and $\widehat{minS}$ contains only minimal infrequent itemsets. However, they are not guaranteed to be duals of each other. `FlipFlop` thus applies the dual operators to construct a new $CS = (\top(\widehat{maxL}) - \widehat{minS}) \cup (\bot(\widehat{minS}) - \widehat{maxL})$. If the duality condition does not hold, CS would be non-empty. The database is then scanned again to find the supports of the itemsets in CS. The result of the counting is used to update $\widehat{maxL}$ and $\widehat{minS}$. The process repeats until CS becomes empty. Figure 5 shows the algorithm `FlipFlop`.

4 Experiments and Results

To evaluate the performance of `FlipFlop`, we performed extensive experiments. Our goals are to study the amount of I/O saving that can be achieved by `FlipFlop` over other algorithms, and how we can get a good guess of $maxL$ by sampling. We will also compare the algorithms in terms of their CPU usage.

4.1 Synthetic data generation

In the experiments, we followed the approach of [3] and [8] and used synthetic data as the test databases. Due to space limitations, readers are referred to [8] for more details. Table 1 shows the parameter setting of our baseline experiment.

4.2 Mining a Sample

We ran a set of simulation experiments to study how sampling should be done to get a good estimate of $maxL$. In the experiments, a synthetic database was generated. A fraction f of transactions were extracted from the database as samples. We then mined the samples using `Apriori`. The maximals of the resulting frequent itemsets discovered were then used as $\widehat{maxL}$ for `FlipFlop`. We repeated the experiment a number of times, each with a different set of samples. For each experiment, we recorded the I/O cost spent and the number of itemsets whose

	Sample size					
	1/2	1/4	1/8	1/16	1/32	1/128
(a) I/O cost						
Mining the sample	4.59	2.25	1.12	0.60	0.30	0.08
`FlipFlop`	2.33	3.17	3.17	4.07	4.33	5.00
`FlipFlop` (Include sampling)	6.92	5.42	4.29	4.67	4.63	5.08
`Apriori`	9					
(b) Itemset counting cost						
Mining the sample	113,460	57,019	28,258	23,242	7,154	1,854
`FlipFlop`	228,996	232,041	233,387	236,945	242,197	257,920
`FlipFlop` (Include sampling)	342,456	289,060	261,645	260,187	249,351	259,774
`Apriori`	224,480					

Table 2. Performance of `FlipFlop` versus sample size ($\rho_s = 1\%$)

supports were counted. These costs included those spent on mining the samples by `Apriori` and on mining the database by `FlipFlop`, and they were calculated by the following expressions. The recorded costs were averaged and compared with those of `Apriori` when `Apriori` was applied to the database directly.

$$\text{Total no. of database scans} = \text{No. of database scans needed by } \texttt{FlipFlop} + (\text{No. of database scans needed by mining the sample} \times f) \quad (1)$$

$$\text{Total no. of itemsets counted} = \text{No. of itemsets counted in } \texttt{FlipFlop} + (\text{No. of itemsets counted in the sample} \times f) \quad (2)$$

Table 2 shows the result of a typical experiment. In the experiment, the support threshold (ρ_s) was set to 1%. In the table, the rows labeled "Mining the sample" shows the costs of applying `Apriori` on the samples; the rows labeled "`FlipFlop`" shows the costs of executing `FlipFlop` when it is supplied with the $\widehat{maxL}$ found from mining the sample; the rows labeled "`FlipFlop` (include sampling)" shows the total costs of the whole mining process.

From Table 2, we see that the larger the sample size is, the fewer I/O passes `FlipFlop` requires. For example, when $f = 1/2$, `FlipFlop` only takes 2.33 passes to complete. This is because the $\widehat{maxL}$ suggested by mining such a big sample is very close to the real $maxL$. This benefits `FlipFlop` greatly. However, the costs of mining the sample also increases with f, hence increasing the sample size to large values become counter-productive. From Table 2, we see that picking an f in the range (1/32 to 1/8) gives a very good I/O performance: the total I/O cost required (4.29 to 4.63 passes) is only about half of what `Apriori` needs.

The CPU requirement of the algorithms can be measured by the itemset counting cost. Similar to the case of I/O cost, a smaller sample size causes less amount of counting performed in the mine-the-sample phase while more counting effort is spent in the `FlipFlop` phase. Again, a smaller sample gives a less accurate estimate of $maxL$, and as a result, some infrequent itemsets are

	ρ_{s_sample}			
	1.0%	0.9%	0.8%	0.75%
I/O cost	4.67	3.92	2.80	2.17
Itemset counting cost	260,187	269,793	293,389	310,741

Table 3. Performance of `FlipFlop` (including sampling) versus ρ_{s_sample}

counted by `FlipFlop` superfluously. From Table 2, we see that picking $f = 1/32$ causes the least amount of total counting effort (249,351). Comparing with the counting cost of Apriori (224,480), `FlipFlop` requires about 10% more CPU time. This CPU overhead, however, brings us an I/O cost reduction of 50%.

In [9], it is shown that lowering the support threshold when mining the samples improves the accuracy of the estimate. We therefore re-ran our experiments with a smaller support threshold ρ_{s_sample} when mining the sample. Table 3 shows the performance of the algorithm with different values of ρ_{s_sample}. In this experiment, the sample size was fixed at 1/16 of the whole database.

From the table, we see that setting a smaller value of ρ_{s_sample} dramatically reduces the I/O cost. For example, when $\rho_{s\_sample} = 0.75\%$, the I/O cost is only 2.17, or a 76% saving when compared with `Apriori`. This I/O cost reduction, however, comes from the expense of a higher itemset counting cost. This is because by mining the sample with a small support threshold, many itemsets are considered frequent in the mine-the-sample phase. As a result, the initial counting set *CS* (line 2, Figure 5) is large. A large *CS* is a mixed blessing: it causes more itemsets to be counted but allows more information be obtained when the supports of these itemsets are determined by scanning the database. If one wants to trade CPU for a better I/O efficiency, especially if the system has a slow storage component, `FlipFlop` can provide the flexibility by means of tuning ρ_{s_sample}.

4.3 Other I/O Efficient Algorithms

Finally, we compare the performance of `FlipFlop` with other I/O efficient algorithms: `DIC` and `Pincer-Search`. In this experiment, we applied the two algorithms on the synthetic database and recorded their I/O costs under various values of the support threshold (ρ_s). Table 4 shows the result. Two rows of numbers for `FlipFlop` are shown. One corresponds to $\rho_{s\_sample} = \rho_s$, and the other (labeled "`FlipFlop` (0.75 ρ_s)") corresponds to $\rho_{s\_sample} = 0.75 \times \rho_s$. For both cases, the I/O costs include those spent in mining the samples.

From Table 4, we see that `FlipFlop` outperforms the others in terms of I/O performance. For example, when $\rho_s = \rho_{s\_sample} = 0.75\%$, a 64% saving in I/O cost (comparing with `Apriori`) was obtained. The reduction is further improved to 78% when a 3/4 support threshold is used in mining the sample. While `DIC` and `Pincer-Search` also performed better than `Apriori` in I/O cost, the savings gained were not as much as that of `FlipFlop`. However, we remark that, in terms of CPU requirement, `FlipFlop` is slightly less efficient than others.

Algorithm	ρ_s			
	1.0%	0.75%	0.5%	0.3%
`Apriori`	9	11	11	11
`DIC`	7.38	7.41	8.20	8.31
`Pincer-Search`	9	8	7	9
`FlipFlop`	4.67	3.95	4.12	7.97
`FlipFlop` ($0.75\rho_s$)	2.17	2.40	3.05	5.55

Table 4. I/O performance

5 Conclusion

This paper described an algorithm framework for discovering frequent itemsets based on the duality property of the set of maximal frequent itemsets ($maxL$) and the set of minimal infrequent itemsets ($minS$). We proved a number of observations governed by the duality property. Based on the observations, we propose efficient algorithms that implement the dual operators. These operators allow us to derive a new algorithm, `FlipFlop` for finding large itemsets. We argued that `FlipFlop` is particular efficient if a good "guess" of $maxL$ is available.

To obtain a good estimate of $maxL$, sampling techniques can be applied. We showed that a small sample is usually sufficient to generate an approximate $maxL$ with a fairly good accuracy. We showed that our algorithm can achieve a significant reduction in I/O cost when compared with other algorithms like `Apriori`, `DIC`, and `Pincer-Search`. The significant I/O improvement is obtained at the expense of a mild increase in CPU overhead. `FlipFlop` is thus an efficient and practical algorithm for mining association rules.

References

1. Charu C. Aggarwal and Philip S. Yu. Data mining techniques for associations, clustering and classification. In *Proc. of the 3rd PAKDD Conference*, Beijing, 1999.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD*, Washington, D.C., May 1993.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th VLDB Conference*, Santiago, Chile, 1994.
4. Alex A. Freitas. On objective measures of rule surprisingness. In *Proc. of the 2nd PKDD Conference*, Nantes, France, September 1998.
5. Christian Hidber. Online association rule mining. In *Proc. of ACM SIGMOD International Conference on Management of Data*, Philadephia, May 1999.
6. Dao-I Lin and Zvi M. Kedem. Pincer-search: A new algorithm for discovering the maximum frequent set. In *Proc. of the 6th EDBT Conference*, 1998.
7. K.K. Loo, C.L. Yip, B. Kao, and D. Cheung. Exploiting the duality of maximal frequent itemsets and minimal infrequent itemsets for I/O efficient association rule mining. Technical report TR-2000-03, Dept. of CSIS, HKU, 2000.
8. Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. ACM SIGMOD*, California, 1995.
9. Hannu Toivonen. Sampling large databases for association rules. In *Proc. of the 22th VLDB Conference*, Bombay, India, September 1996.

Data Mining Using Query Flocks with Views

Meliha Yetisgen and I. Hakki Toroslu

Dept. of Computer Eng. METU
Ankara, Turkey

Abstract. Data Mining is the process of finding trends and patterns in large data. Association rule mining become one of the most important techniques for extracting useful information such as regularities in the historical data. Query flocks extends the concept of association rule mining with a "generate-and-test" model for many different kind of patterns. This paper further extends the query flocks with view definitions. Also, a new data mining architecture simply compiles the query flocks from datalog to SQL. On this architecture, optimizations suitable for the extended query flocks are introduced. The prototype of the system is developed on a commercial database environment. Advantages of the new design and the extension to the query flocks, together with the optimizations, are also presented.

1 Introduction

Over the past decade, many organizations and research groups have began to routinely capture huge volumes of data describing their operations, products, and customers. The field of *Data Mining* addresses the question how best to use this historical data to discover regularities and improve the process of making decisions, such as analyzing medical outcomes, detecting credit card fraud, predicting the personal interests of web users, and optimizing manufacturing processes.

Market Basket Analysis is a simple but important example to predict customer purchase behaviour. A market basket data is a collection of items purchased by a customer in a single customer transaction. The analysis of basket data can be used to decide what to put on sale, how to put design placement of goods on shelves, how to decide sales promotions, and many other decisions that are taken in order to increase the profit in a supermarket. To solve this problem, the main step is to find sets of items that are associated. The fact of their association is called an *Association Rule* [1–3, 5, 6, 8]. Association rules are a class of simple but powerful regularities in data. The rules have the form $A \Rightarrow B$, where in the case of market basket analysis both A and B are sets of items. If every item in A is purchased then it is likely that items in B will also be purchased. There are two important measures for an association rule:

- *Support:* All the items that appear in $(A \cup B)$ must appear in many baskets.
- *Confidence:* The probability of one item in B to be in the basket, given that the others in A are in the basket, must be high.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 720–729, 2000.

To speed up the search for high support sets *A-priory Property* [1–3, 6–8] is used.

A-priory Property: Every subset of a frequent item-set must also be a frequent item-set.

On the other hand, *Query Flock* that was presented in [8], generalizes the a-priory property for larger class of problems. The idea of flocks is performing complex data analysis tasks on relational database systems. The setting for a query flock systems is:

- A language to express parameterized queries,
- A language to express filter conditions about the results of the query.

Datalog, a relational query language inspired by Prolog [5, 8], is selected as the language for expressing parameterized queries. One of the important reasons for this selection is the set of options for adapting the a-priory trick to arbitrary flocks is most easily expressed in datalog. Given these two languages a query flock can be specified by designating:

- One or more predicates that represent data as relations,
- A set of parameters whose names beginning with $,
- A query expressed in a language using parameters as constant,
- A filter condition that the results of the query must satisfy in order for a given assignment of values to be acceptable.

The meaning of a query flock is a set of tuples that represent acceptable assignments of values for the parameters. The acceptable parameter assignments are determined by trying all such assignments in the query, evaluating the query, and seeing whether the result passes the filter test. Market basket analysis for pairs of items $\$I1$ and $\$I2$ that satisfy the filter condition can be given as:

```
QUERY:
    ans1(B):-baskets(B,$I1) AND
             baskets(B,$I2) AND
             $I1 < $I2
    ans2(B):-baskets(B,$I1)
FILTER1:
    COUNT(ans1.B) > N
FILTER2:
   COUNT(ans1.B) > COUNT(ans2.B) * c.
```

In the first filter condition, support is used for finding pairs of items that appear in at least N baskets. On the other hand in the second filter condition, confidence is represented in order to find pairs of items $\$I1$ and $\$I2$ such that the confidence of $\$I2$ given $\$I1$ is at least c [8]. It is important to realize that a query flock is a query about its parameters. The parameter values are the ones that

satisfy a minimum support or confidence condition and give knowledge about possible related items.

Query Flock Compiler, a tightly coupled approach with the database, was introduced in [7], to process the flocks. The main advantage of choosing a tightly coupled approach is to enable to take the full capabilities of database technology. The data is stored in the database and all query processing is done locally at the database. We add views in parameterized query definitions and design a view evaluator that sits on the top of external optimizer. Our new architecture for query flock compiler can be seen in Figure 1.

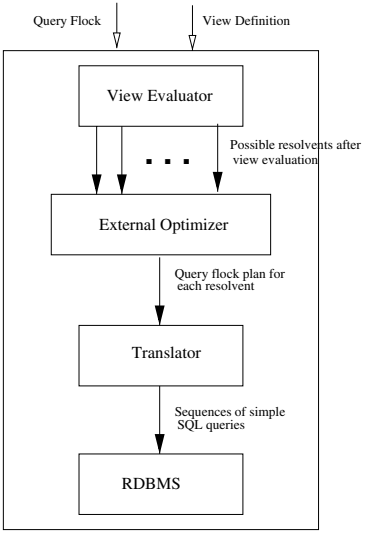


Fig. 1. Query Flock Compiler Architecture

The next sections give the design details of the architecture by discussing the ideas taken from recent researches and new introduced ideas with many simple conceptual examples. Sections 2, 3 and 4 cover View Evaluator, External Optimizer, and Translator, respectively. Section 5 includes implementation details and performance results of sample runs. Finally in Section 6 the paper is concluded.

2 View Evaluator

As mentioned before, in recent researches datalog was selected as query flock language. In datalog there are two ways relations can be defined. A predicate whose relation is stored in the database is called *extensional database* (EDB) relation, while one defined in logical rules is called *intentional database* (IDB) relation. We use the term *view* for IDB relations since the capability to create

views in models like relational model is somewhat analogous to the ability in datalog to define IDB relations and we assume each predicate symbol either denotes an EDB relation or an IDB relation but not both.

If the query part of the flock includes IDB relations these relations have to be evaluated. View evaluator takes clauses of IDB relations and query part of the flock as inputs; after processing these inputs, it returns all possible queries whose relations are all EDB relations. We use mainly the ideas in [4] to make view evaluation. The clauses of IDB relations are represented in the form of a *Connection Graph*. In recent works on flocks, the parameterized queries are formed with only EDB relations [7, 8]. The below example shows how views are included in a flock definition and the usage of connection graphs for finding possible resolvents for such systems.

Example 1.
```
IDB Relations:
    treatments(Patient,Medicine) \\Patient is treated with the
                                 \\medicine.
    diagnoses(Patient,Disease)   \\Patient is diagnosed with the
                                 \\disease.
EDB Relations:
    cures(Medicine,Disease)   \\ Medicine is used for the disease.
    exhibits(Patient,Symptom) \\ Patient exhibits the symptom.
    causes(Disease,Symptom)   \\ Disease causes the symptom.
    not_allergic(Patient,Medicine) \\ Patient is not allergic
                                   \\ with the medicine.
Rules:
    treatments(Patient,Medicine):-cures(Medicine,Disease) &
                                  diagnoses(Patient,Disease) (1)
    treatments(Patient,"abc"):-not_allergic(Patient,"abc") &
                               cures("abc",Disease) &
                               diagnoses(Patient,Disease)    (2)
    diagnoses(Patient,Disease):-exhibits(Patient,Symptom) &
                                causes(Disease,Symptom)      (3)
```

Connection Graph for IDB relations, *diagnoses* and *treatments*, can be seen in Figure 2.

By using the following flock [8], we search for unexplained side effects on patients who have "*asthma*", in other words, we want to find the symptoms $\$S$ and medicines $\$M$ where there are many patients that exhibit the symptom and are taking the medicine, yet the patient's disease ("*asthma*" is used as a constant value for disease) does not explain the symptom.

```
QUERY:
    ans(P):-exhibits(P,$S) AND
            treatments(P,$M) AND
            diagnoses(P,"asthma") AND
            NOT causes("asthma",$S)
FILTER:
    COUNT(ans.P) > N
```

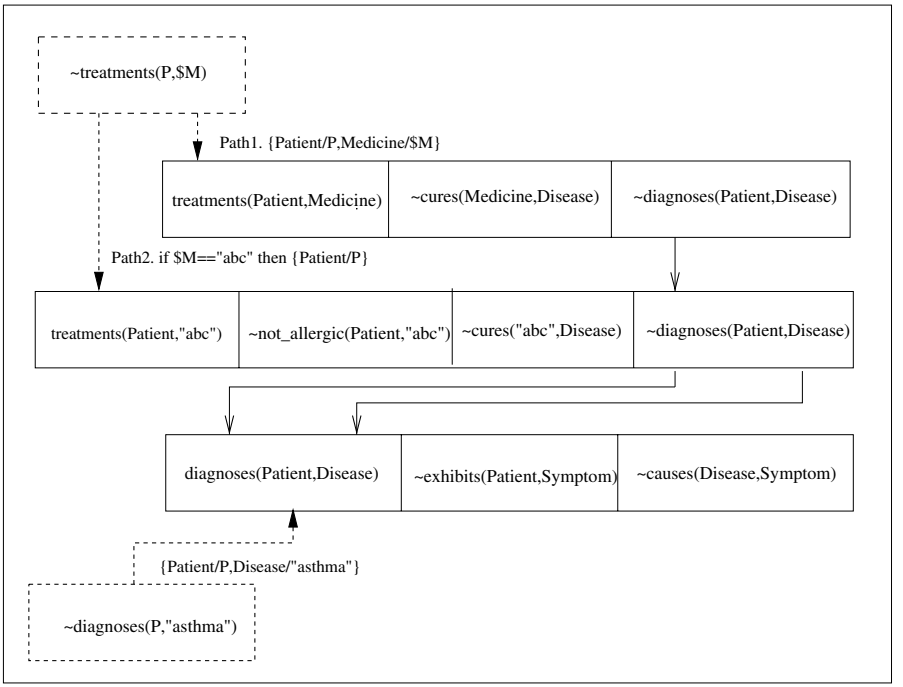


Fig. 2. Connection Graph for Example 1

The query part contains IDB relations: *treatments* and *diagnoses*. For resolving *treatments*, the query is $treatments(P, \$M)$ and as can be seen from the graph for $treatments(P, \$M)$ there are two paths.

First Path: From (1) we get

```
treatments(P,$M):-cures($M,Disease) & diagnoses(P,Disease).     (4)
```

But (4) still contains an IDB relation, *diagnoses*. From (3) and (4) we get

```
treatments(P,$M):-cures($M,Disease) & exhibits(P,Symptom) &
                  causes(Disease,Symptom).                      (5)
```

Second Path: Here parameter $\$M$ is unified with constant value "*abc*". To achieve this unification, we introduce an *if clause*.
if \$M == "abc" then from (2) we get

```
treatments(P,$M):-not_allergic(P,"abc") & cures("abc",Disease) &
                  diagnoses(P,Disease).                         (6)
```

(6) still contains an IDB relation, *diagnoses*. From (3) and (6) we get

```
treatments(P,$M):-not_allergic(P,"abc") & cures("abc",Disease) &
                 exhibits(P,Symptom) & causes(Disease,Symptom).(7)
```

On the other hand for *diagnoses*, the query is *diagnoses*(P,"*asthma*") and from (3) we get

```
diagnoses(P,"asthma"):-exhibits(P,Symptom) &
                       causes("asthma",Symptom).          (8)
```

After resolving IDB relations *treatments* and *diagnoses*, from (5) and (8) we get

```
QUERY:
   ans(P):-exhibits(P,$S) AND cures($M,Disease) AND
           exhibits(P,Symptom) AND causes(Disease,Symptom) AND
           exhibits(P,Symptom) AND causes("asthma",Symptom) AND
           NOT causes("asthma",$S)
FILTER:
    COUNT(ans.P) > N,
```

and from (7) and (8) we get

```
QUERY:
   ans(P):-exhibits(P,$S) AND not_alergic(P,"abc") AND
           cures("abc",Disease) AND exhibits(P,Symptom) AND
           causes(Disease,Symptom) AND exhibits(P,Symptom) AND
           causes("asthma",Symptom) AND NOT causes("asthma",$S)
FILTER:
   COUNT(ans.P) > N.
```

A recursive intension occurs in a connection graph as a particular kind of cycle called potential recursive loop [4], or simply a loop. The algorithm of view evaluator accepts only nonrecursive IDB relation clauses (acyclic connection graphs) as inputs, but as a future work, recursive IDB relation evaluation will be added to our view evaluator.

3 External Optimizer

The databases under consideration are very large ones, as a result, direct translation of flocks to SQL will be very inefficient due to the time it takes. In order to overcome this difficulty *QueryPlans* are proposed as a solution in [7]. A query plan can be defined as a pre-optimizer of complex mining queries and then used to feed the sequence of smaller simpler queries to the query optimizer at the DBMS. A query flock plan is a sequence of the following three operations;

1. Materialization of an auxiliary relation,
2. Reduction of a base relation,
3. Computation of the final result.

We extend the idea of query plans [7] for rules with IDB relations. We find query plans for all possible resolvent flocks after view evaluation of initial query flock with IDB relations and propose a simple reordering algorithm among the predicates of each query flock plan for optimization. The next example is given to summarize the main ideas and advantages of query flock plans.

Example 2. In the previous example we have eliminated the views from flock definition and found two possible resolvents. In this example we continue with the first resolvent query flock as input to the external optimizer.

Before discussing about the example we want to give the meaning of *Auxiliary Relation*. An auxiliary relation is a relation over a subset of parameters of a query flock and contains candidate values for the given subset of parameters. In other words all parameter values that satisfy the filter condition are contained in the auxiliary relation, or any value that is not included in the auxiliary relation is guaranteed not satisfy the filter condition. A possible auxiliary relation for our flocks is given below. The auxiliary relation aux_S can be defined for parameter $\$S$ as the result of the following query flock. Relation aux_S will contain all values for $\$S$ that satisfy the filter condition.

```
QUERY:
    ans(P):-exhibits(P,$S)
FILTER:
   COUNT(ans.P) > N
```

And the auxiliary relation aux_M can be defined for parameter $\$M$ as the result of the following query flock.

```
QUERY:
   ans(P):-cures($M,Disease) AND exhibits(P,Symptom) AND
           causes(Disease,Symptom)
FILTER:
   COUNT(ans.P) > N
```

Materialization of an auxiliary relation is actually a query flock, such as in the previous relation, that is meant to be executed at the RDBMS. Auxiliary relations are used to simplify the original query flock. Relation aux_S can be joined with $exhibits(P, \$S)$ relatively quickly and most probably the result of this join will be a smaller relation and subsequent join steps take less time than they would in the original query flock.

Reduction of a base relation is a simple semi-join of a base relation with one or more previously materialized auxiliary relation and the result replaces the base relation. Here the important point is relations that contain parameters as arguments are reduced by auxiliary relations. For our case if $exhibits(P, \$S)$, $causes(\text{"}asthma\text{"}, \$S)$ and $cures(\$M, Disease)$ are reduced with $aux\_S(\$S)$ and $aux\_M(\$M)$, respectively, we get

```
new_exhibits(P,$S):-exhibits(P,$S) & aux_S($S),
new_causes("asthma",$S):-causes("asthma",$S) & aux_S($S),
new_cures($M,Disease):-cures($M,Disease) & aux_M($M).
```

After reduction phase for parameters $\$S$ and $\$M$, for the input flock, we get

```
QUERY:
   ans(P):-new_exhibits(P,$S) AND new_cures($M,Disease) AND
           exhibits(P,Symptom) AND causes(Disease,Symptom) AND
           exhibits(P,Symptom) AND causes("asthma",Symptom) AND
           NOT new_causes("asthma",$S)
FILTER:
   COUNT(ans.P) > N.
```

Algorithm for External Optimizer that generates query flock plans is taken from [7] with some minor changes. For further information and performance results of External Optimizer, see [7].

In order to improve the performance of query flock plans we propose a simple reordering algorithm that allows evaluation of reduced relations before other relations of the flock query. As an example, if we reorder the relations of the flock that was generated in Example 2, we get

```
QUERY:
   ans(P):-new_exhibits(P,$S) AND new_cures($M,Disease) AND
           NOT new_causes("asthma",$S) AND
           causes(Disease,Symptom) AND exhibits(P,Symptom) AND
           causes("asthma",Symptom) AND exhibits(P,Symptom)
FILTER:
   COUNT(ans.P) > N.
```

4 Translator

The final step for Query Flock Compiler is to translate datalog rules to SQL statements. The input to the translator is a rule or a query flock, consists of subgoals, $S1, \ldots, Sn$. For each $Si = pi(Ai1, \ldots, Aik)$ with an ordinary predicate pi, there is a relation $Ri(Di1, \ldots, Dik)$ already computed, where the A's are arguments, either variables or constants and D's are the names of the columns of computed relation. Parameters are covered in variables. As mentioned before we are dealing with only non-recursive safe datalog rules. The main steps of the translation algorithm are:

1. If the input is a rule, SELECT clause of the SQL statement will contain one column name for each argument of the head predicate, or if the input is a query flock, SELECT clause of the SQL statement will contain one column name for each parameter.
2. For both input types, the FROM clause contains one relation per-occurrence of a predicate in the body of the rule.
3. For both input types, the conditions in the WHERE clause are a conjunction of the followings:
   - For each constant in an ordinary predicate Si in the body (ie. $Si(X,' a')$), a condition which equates the constant with the corresponding attribute,

- For each occurrences of the same variables in an ordinary predicate in the body (ie. $Si(X, Y, X)$), the necessary number of conditions that signify the equality of the attributes corresponding to these variables,
- For each occurrences of the same variables in the ordinary predicates in the body (ie. $Si(X, Y), Sj(X, Z), Sk(T, X)$), the necessary number of conditions that signify the equality of the attributes corresponding to these variables. For the given example the variable X appears three times in the body, we will have two conditions that indicate that the three attributes corresponding to the three occurrences are equal.

4. If the input is a query flock, GROUP BY clause contains the attributes corresponding to the parameters. If the input is a query flock, the HAVING COUNT clause contains the filter condition.

Example 3. The flock for auxiliary relation *aux_S* in Example 2 is translated to SQL as

```
aux_S(Symptom) AS
  SELECT r1.Symptom
  FROM exhibits r1
  GROUP BY r1.symptm
  HAVING COUNT(r1.Symptom) > N.
```

The reduction step for relation *exhibits* is translated to SQL as

```
new_exhibits(Patient,Symptom) AS
  SELECT r1.Patient, r2.Symptom
  FROM exhibits r1, aux_S r2
  WHERE r1.Symptom = r2.Symptom.
```

5 Implementation Details and Sample Runs

The inputs to the Query Flock Compiler are given as text files. These files include query flock definition, clauses of views (IDB relations), and attribute names of base relations. The compiler code is implemented in C and SQL queries are run over WindowsNT ORACLE 8.0 standard database.

Our experiments are based on the example flock that searches for unexplained side effects. In Example 2, we had found two possible resolvent for the initial flock as a result of view evaluation. We used the first resolvent flock and its query plan in our experiments and compared the performance of them in terms of support and table sizes.

Case 1: Exhibits relation contains 50.000 tuples, cures and causes relations contain 2500 tuples (Figure 3).

Case 2: Exhibits relation contains 25.000 tuples, cures and causes relations contain 1500 tuples (Figure 3).

As can be seen from the graphs query plan performs better than standard execution of query flock and query plan total execution time decreases as the support increases. This is a trivial result of reducing base relations with related auxiliary relations in external optimizer phase of query flock compiler.

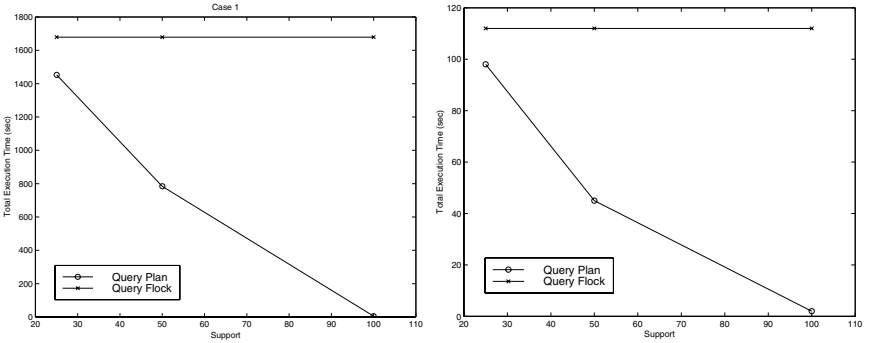


Fig. 3. Performance Results for Case 1 and Case 2

6 Conclusion

In this paper we presented a new data mining architecture that is tightly coupled with the RDBMS. In our architecture, the idea of query flocks and optimization techniques on flock queries that were proposed in previous researches, were taken as base. We added view definitions in flock queries and presented an algorithm for evaluating them. We also reported performance results for different table sizes. The view definitions under consideration were non-recursive, but as a future work we will add new features that evaluate recursive view definitions.

References

1. R. Agrawal. Fast Algorithms for Mining Association Rules. *Proceedings of the* 20^{th} *VLDB Conference*, pages 487–499, 1994.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining Assocation Rules between Sets of Items in Large Databases. *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.
3. R. Groth. *Data Mining - Building Competitive Advantage*. Prentice Hall, 1999.
4. L. J. Henschen. On Compiling Queries in Recursive First-Order Databases. *Journal of the Associations for Computing Machinery*, 31(1):47–85, 1984.
5. R. Ramakrishnan. *Database Management Systems*. McGraw-Hill, 1997.
6. R. Srikant and R.Agrawal. Mining Generalized Association Rules. *Proceedings of the* 21^{th} *VLDB Conference*, 1995.
7. D. Tsur, S. Nestorov, and A. Rosenthal. Integrating Data Mining with Relational DBMS: A Tightly Coupled Approach. *Proceedings of* 4^{th} *Workshop on Next Generation Information Technologies and Systems NGITS'99*, 1999.
8. D. Tsur, J.D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query flocks: A Generalization of Association-Rule Mining. *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1998.

A KDD Experience Factory: Using Textual CBR for Reusing Lessons Learned

Kai Bartlmae and Carsten Lanquillon

DaimlerChrysler AG
Research & Technology 3
89013 Ulm, Germany
kai.bartlmae@daimlerchrysler.com
carsten.lanquillon@daimlerchrysler.com

Abstract. We introduce an integrated framework for Knowledge Discovery in Databases (KDD) and Knowledge Management and show how Knowledge Management can complement KDD. Specifically, we examine methods how to improve the knowledge intensive and weak structured process of KDD through the use of an experience factory using the method of experience packaging and case based reasoning (CBR).
This paper investigates how knowledge contained in the textual components of experience packages can be used to improve the retrieval of lessons learned in KDD. We add textual CBR techniques to our CBR approach in order to improve the case retrieval mechanism of the experience factory. Our technique exploits domain-specific knowledge contained in the textual parts of the packages to find better reuse candidates of lessons learned in KDD.

Keywords: Knowledge discovery in databases, case base reasoning, experience reuse, information retrieval, knowledge management.

1 Introduction

In the last years, many technologies and organizational methods have been created in order to support organizations conducting KDD. The use of process models give an explicit conceptional representation of phases or tasks to be conducted making the process of KDD more transparent and communicable. The definition of generic KDD processes such as CRISP-DM [12] or the approaches described in [9, 10, 13] has been a major step towards making KDD more structured and applicable in a business environment with larger teams. Note that these process models have to be tailored, mapped or revised on the specific application domain and problem characteristics in to be applicable.[1]

In this paper, a framework is put on what KDD steps should be taken, giving guidelines for a KDD initiative and setting quality standards on the process steps, i.e. on deliverables. We consider experience to be the key factor throughout the process of KDD. Moreover, through continuous feedback on projects and continuous learning about a domain, domain specific KDD processes change incrementally and evolutionary.

[1] In CRISP-DM, this is called "mapping"

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 730–740, 2000.

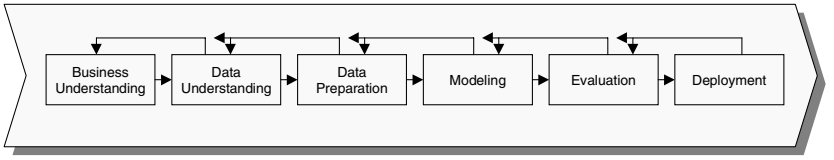


Fig. 1. The Crisp Reference Model: Phases, Generic Tasks (bold) and Outputs (italic) of the CRISP-DM Reference Model [12]

It has been proposed that experiential knowledge is an important source of knowledge that can contribute to the ability of problem solving [11]. Experience can be used to respond to new situations in a similar way and to avoid similar failures. This is a form of continuous learning based on which an organization can improve. We introduced a method [4] that adds experience to the process of mapping Data Mining (DM) processes to applicable KDD tasks and executing them. Using a former solution of a similar KDD problem as a basis will decrease the time and cost spent on the current problem. Further, the creation of an organization wide knowledge infrastructure including past experiences promotes the use of KDD technology in practice.

The remainder of the paper is organized as follows. In a first step we explain why it is important to collect and reuse experience and knowledge about KDD and introduce our KDD experience factory approach. Further we show how larger textual components of the experience packages can be incorporated in our CBR approach in the case of KDD lessons learned.

2 Application of KDD Processes

By a KDD process model we understand a general framework for creating and using knowledge derived from data. A generic strategy to do so is the Cross Industry Standard Process of Data Mining (CRISP-DM) process model [12] which uses KDD to solve a particular business problem. CRISP-DM formulates different phases, broken down to tasks and process instances. It specifies general steps that taken together help structuring KDD and help to fulfill minimal quality standards. The process is divided into the phases business understanding, data understanding, data preparation, modeling, evaluation, and deployment. See Figure 1 for an overview of CRISP-DM. Since the framework is generic, more information is needed in order to solve specific business problems. In CRISP-DM the process of deriving a KDD process from the model is called mapping. From the given KDD phases different layers of abstraction, from *KDD phases, generic tasks, specialized tasks* to *process instances*, are derived and executed (see Figure 2). In this framework, information about mapping and execution has to come from KDD specialists and business analysts. Hence, the difficult part is to apply a mapping from the generic model to concrete KDD tasks.

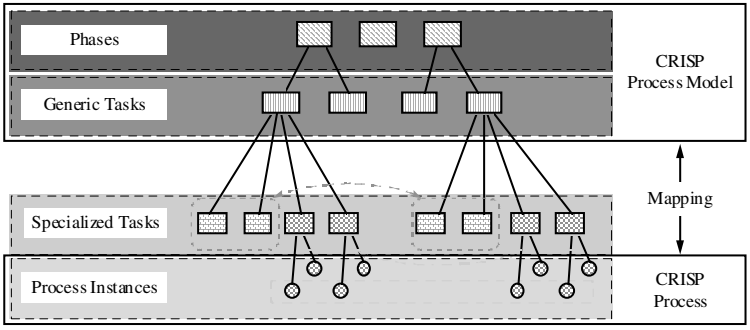


Fig. 2. The Crisp Mapping Model: Four Level Breakdown of the CRISP-DM Methodology. At the specialized task level, the same colored boxes belong to the same project: Since CRISP-DM is generic, different projects processes can be derived (From [12]).

3 Knowledge Enabled CRISP-DM

We see KDD as a knowledge intensive and weak structured process. Moreover, in a larger business environment, KDD will have the characteristics of project work: separate teams are involved in solving changing problems. In this sense, the collection of experience and the transfer of knowledge from one project team to another is a key issue. Persons involved in KDD can learn from this experience. Further, the process of mapping the process model of CRISP-DM onto a KDD process and executing it is highly context and hence experience depended. However, CRISP-DM user guides give only rudimentary support when it comes to *how* this mapping can be done and *where* the detailed information to perform KDD has to come from.

In order to perform process mappings and execute process instances, lessons learned of former CRISP-DM processes can be used, but only if experience is secured in a way that allows future reuse (see Figure 3). CRISP-DM requires the project member to document KDD results and general experience which, however, is not explicitly reused. Therefore knowledge on how to perform KDD is only added when new reference and user guides[2] are released.

We therefore follow a different strategy for project support on KDD. In our opinion, an experience and knowledge management process incorporating CRISP-DM promotes good KDD practice and prevents the repetition of mistakes. This means that there is a need for systematic collection, dissemination and reuse of experiences made in KDD processes, especially in a corporate environment in which many different teams work on similar tasks and can share their knowledge.

Systematic knowledge creation, collection, organization, dissemination and reuse provides a new way to support the KDD process model CRISP-DM. In this sense, we embedded CRISP-DM in an experience management approach to support the mapping and execution of KDD processes. We identified different types of knowledge that can

[2] I.e. KDD for predictive modeling in customer relationship management.

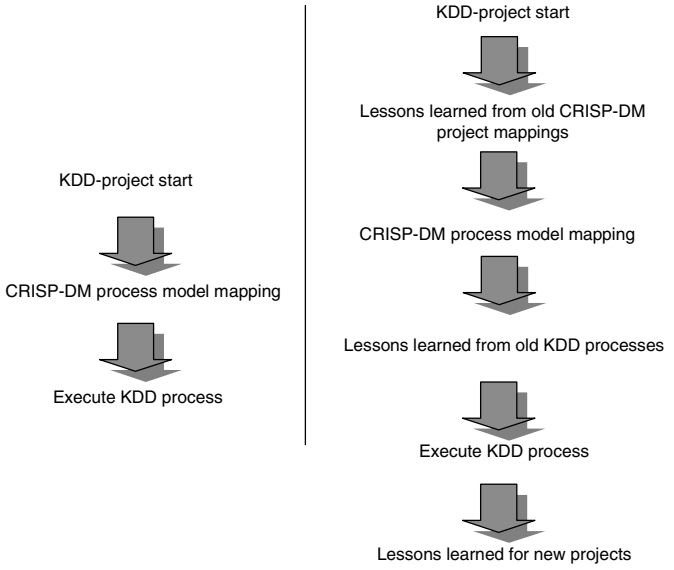


Fig. 3. From KDD processes to the integration of lessons learned in KDD (Derived from [21])

improve KDD processes and provided a way to reuse them using an experience factory. Since CRISP-DM is used in all our projects, comparable and reuseable patterns of how to map the CRISP-DM process model onto a specific business problems' KDD task can be derived from the experience of preceding projects.

3.1 Connecting KDD Project Management with Knowledge Management

Knowledge management and organizational learning frameworks are based on a learning cycle. Therefore, we embedded the CRISP-DM process model within a knowledge management model derived from the definitions of [21] and [17]. The knowledge enabled CRISP-DM process model is a synthesis of the CRISP-DM process model interacting with this knowledge management process (see Figure 4). The CRISP-DM process model is extended such that prior experiences are retrieved and reused and important experiences concerning the tasks are documented after each step or after the whole process.

Our KDD knowledge model is devided into three components(See the *knowledge manangement process* in figure 4). The first is the definition of knowledge goals. Next, a realization of a knowledge management process is derved from that, identifying, collecting, disseminating and reusing KDD experience through an experience base. Third, from this realization a valuation is made and used to form new goals.

We use the approach of *Case Based Reasoning* (CBR)[3] for the the realization of the experience management process by means of an experience base in KDD (also see

[3] See [1] for an overview

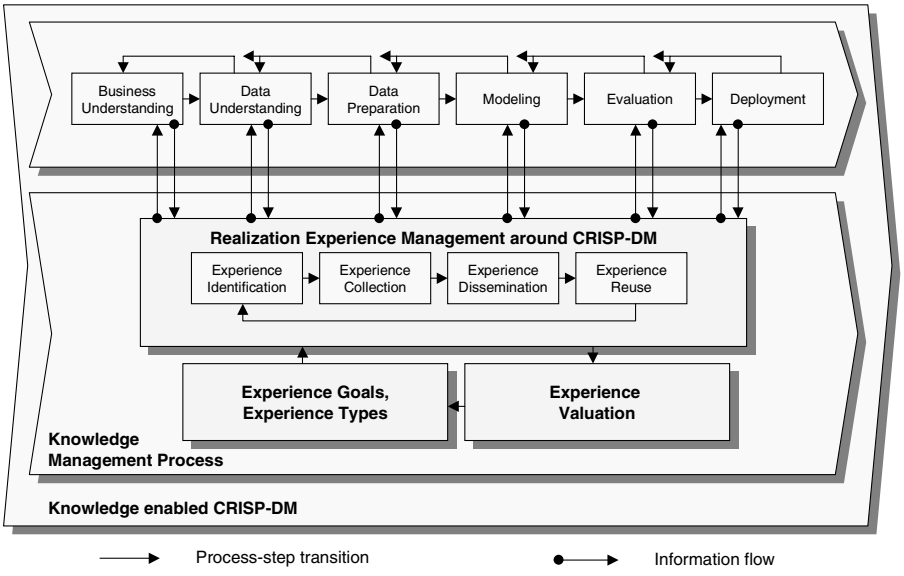


Fig. 4. Reference model for experience management in CRISP-DM

[5]). The CBR approach is based on the idea that new problems can be solved by using similar problems that have been solved before. Thus, it is based on human tendency to reason from cases. All information is stored with the goal of problem solving. Consequently, the purpose of the model is not process and result documentation but the description of reusable cases in form of problem-solution pairs.

For a new problem, the most similar cases as determined by a predefined measure based on text, given attributes, etc., are retrieved from the case base and reused. The newly solved problem is then examined and appropriate cases are stored in the case base. In this paper, cases in form of experience packages are captured and stored in a way that they can be reused best. In this way, incremental learning is an integral part of the reuse process.

3.2 The KDD Experience Factory

By a KDD experience factory we understand a separate organization that captures and distributes different types of knowledge in predefined forms, from lessons learned to data descriptions [4]. Browsing and search mechanisms make it possible to share, discuss and learn through the whole organization, thus catalyzing continuous improvement (see Figure 5). The concept of an experience factory originates in the field of software engineering where it has proven to be successful [7, 2].

The product of such an experience factory is an experience package. Its content and structure varies based upon the kind of experience clustered in it. In our case, the experience of a KDD task is memorized through special KDD packages that are characterized by context attributes and give information on how to execute a KDD step in

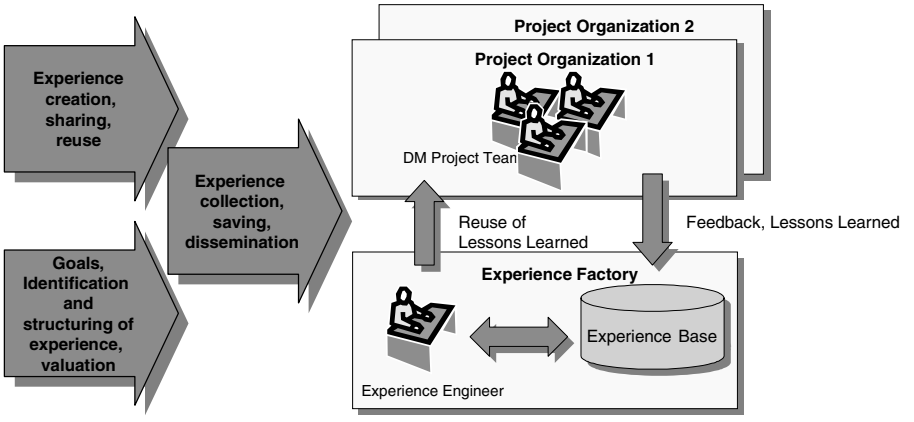


Fig. 5. The model for the management of KDD knowledge through an Experience Factory: Data Mining teams collaborate, subscribe to and capture DM knowledge in order to share experience through an Experience Factory (also see [7])

that context, to solve problems or give lessons learned from past projects. The packages have a structured form: we use a domain model's attributes to characterize each packages context and represent the experience in several attributes or text form fields with references to other artifacts.

We derived ten types of experience packages to be stored in the experience base and disseminated through the factory[4]. Further, we use an object-oriented package model including generalizations such that common attributes are shared by different package types.

4 Using Textual CBR for the Lessons Learned Experience Packages

We use a structured approach for representing cases. For the lessons learned, each of the packages is structured in a description of the problem, a method or solution and information about the rationale and the cause. Further, we use our derived domain model to describe the context for classification of the packages (and for effective retrieval). The domain model consists of over 10 attributes to describe the context of the lessons learned packages, i.e. the KDD phase or task, the application type, the involved objects, the problem class, the applied methods, the used tools, the life cycle of the experience or the specialization of the experience. Through the use of taxonomies, i.e. for KDD phases and tasks, we are able to model dependencies such that the *Data Description task* belongs to the *Data Understanding phase*. Further, we build a domain specific vo-

[4] Lessons learned (management and KDD), process, metadata on datasets used (2), KDD products, KDD methods, i.e. neural nets, experts and skills, formulas, i.e. error or quality measures, other artifacts

cabulary in order to insert keyword attributes that represent the context on a specialized level.

For all package types but the lessons learned we found that using predefined attributes in the sense of a structured CBR approach [6] is sufficient to describe the packages. So far we further used the vocabulary to describe the packages textual components. The keyword concept allows the introduction of additional context descriptions and help the user to identify useful packages. This approach has also been used by other EF implementations [19, 15].

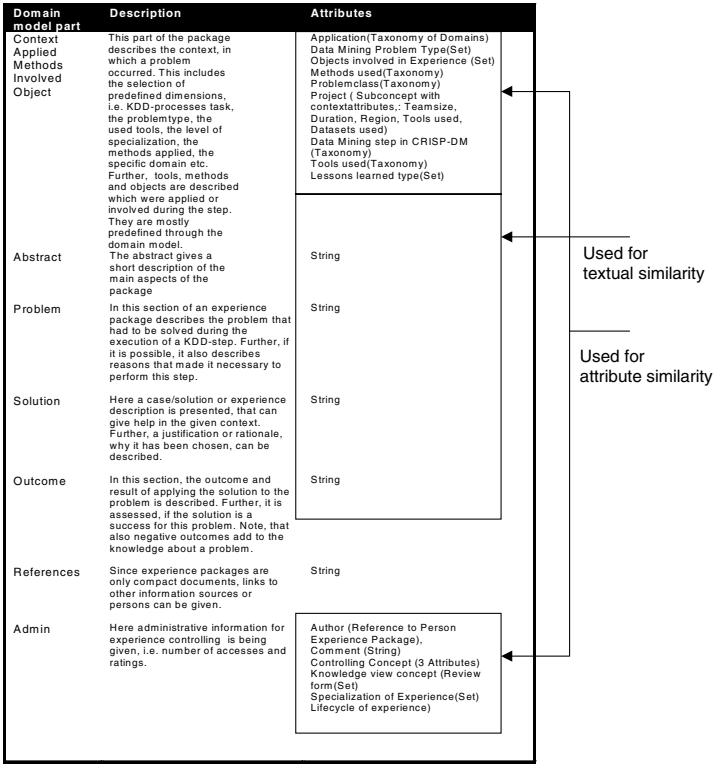


Fig. 6. Model for the representation of KDD lessons learned

In this paper, we follow a different approach. Rather than relying on the experience engineer to find good keywords, we combine our structural CBR approach with a textual CBR technique(tCBR) for the representation of the knowledge of the textual parts (see Figure 6).

Here, we rely on the structured form of the lessons learned and use the textual components to extract *Information Entities (IEs)* about the packages [20]. The knowledge for identifying these IEs of the packages is given by a set of term indicies, thesauri, a product/name-index and a term-generalization-index. Further, for the term indicies and

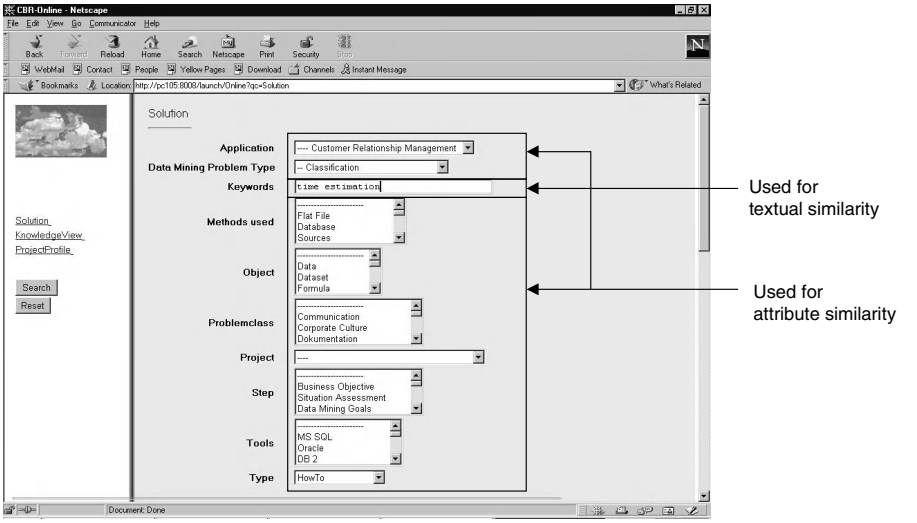


Fig. 7. Querying the experience base for lessons learned. We distinguish an attribute part, we one can make us of a structured domain model and use common CBR similarity measures and the textual part, that makes use of domain-dependent knowledge of the textual parts of the lesson learned.

the thesauri we use a domain-independent and a domain-dependent form. The content of the dictionaries is collected by our domain experts or automatically by parsing KDD related documents.

For retrieving lessons learned we distinguish the attribute part, where we can make use of the structured domain models predefined attributes and their possible values, and the textual part, which makes use of domain-dependent and common knowledge stored in the index-vocabulary, thesauri and term-generalizations.

For the attribute part, we defined for the different attribute types (i.e. integers, symbols, symbol-sets, taxonomies) similarity measures for their values. For the textual parts, a query to the experience base should give results similar to a package, that contains similar expressions in form of the IEs. Here, we measure the similarity *SIM(Q,E)* of a textual query *Q* to an experience package *E* by summing the similarities of each of the IEs of Q and E:

$$\mathrm{SIM}_{\mathrm{tCBR}}(Q, E) = \frac{\sum_{\mathrm{ie}_Q \in Q} \sum_{\mathrm{ie}_E \in E} \mathrm{sim}(\mathrm{ie}_Q, \mathrm{ie}_E)}{|Q|} \tag{1}$$

with ie_Q and ie_E being the IEs in the query and the experience package. The similarity between two IEs is 1 if they are the same and 0 otherwise. The resulting overall similarity $\mathrm{SIM}(Q, E)$ is then calculated as a weighting of the similarities of all attributes and $\mathrm{SIM}_{\mathrm{tCBR}}$. The steps of the calculation are shown in Figure 8.

Before the experience base can be queried, the lessons learned packages' IEs have to be pre-calculated. This is done in an off-line process as shown in Figure 9. In a first step, only IEs found in the lessons learned and the domain-independent term-index are

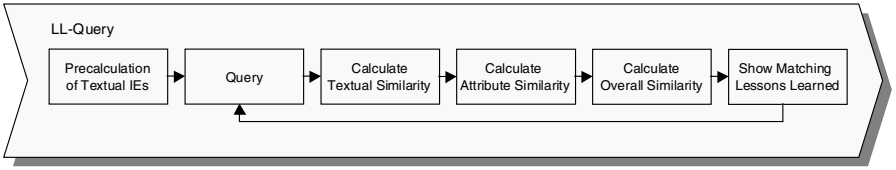


Fig. 8. Steps of queuing the experience base for lessons learned

selected, followed by the selection of KDD/Business dependent terms and phrases. In the next step, product or KDD specific names are inserted and generalizations of the IEs are included. Further, similar IEs of the already selected are included using two thesauri. This all makes it possible that a query matches a package while using different words.

4.1 Implementation

For the implementation of the experience base we use the CBR tool CBR-works from TecInno. The tool supports an object oriented concept model which we made use of for the representation of the different types of experience packages.

Its basic support for textual components is based on keyword searching on the involved field. We implemented a first prototype of our experience base realizing all the different package types using our derived domain model and concepts. The precalculation of the IEs is performed in the case that a new package is checked into the experience base by a separated system fully implemented by FT3/AD. Further, the base is accessible for the users via our department's intranet. Like for all package types one can query via intranet the experience base for all package types by giving the context of a situation, i.e. for lessons learned (see Figure 7). A list of answers is then presented which maximize the similarity of all experience packages to the query according to the domain model.

So far we reviewed four finished KDD projects, three in the domain of KDD in credit risk management and one in predictive modeling and marketing. We collected over 300 experience packages, most belonging to the package type of lessons learned. We parsed KDD-specific documents for a basic domain-dependent and independent term-index. Further, we collected the thesaurus's synonyms and build up the term generalizations. Most parts of the structured CBR-approaches domain model were used in the textual CBRs terms, phrases and generalizations. So far, we use over 2500 keywords, phrases and relations for selection of the IEs.

5 Conclusion

Systematic knowledge creation, capture, organization and use provides a new way to support the KDD process model CRISP-DM. We see the KDD process as a knowledge

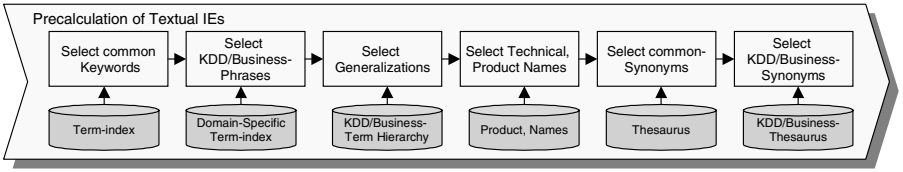


Fig. 9. Steps for precalulating a lessons learned IEs, using terms, thesauri and generalizations

intensive and weak structured process where the agents have to choose on each step from a variety of options. This makes organizational team support an important issue in the case of KDD. At DaimlerChrysler, different KDD teams apply the CRISP-DM process in projects from credit scoring to customer care. Therefore, we identified sources of experiences that can improve KDD processes and showed how experience can be integrated using a CBR based Experience Factory for KDD.

Here we incorporated text retrieval techniques into our mainly attribute based retrieval mechanism. This makes it possible to use the knowledge inherent to the textual components of our lessons learned experience packages. We use index terms, thesauri, product names and generalization relations to find knowledge in these packages and to exploit them in our retrieval mechanism.

In the future, we will constantly adapt the experience base as new experience is gathered and try to integrate the factory within other KDD projects, i.e. credit risk. On the technical side, we are working on topics to improve the similarity measures and differentiate between the different types of knowledge inherent to the found information entities.

References

1. Aamodt, A. Plaza, E.: "Case-based reasoning: Foundational issues, methodological variations, and system approaches". AI-Communications, 7(1), 39-59, 1994.
2. Althoff, K.-D. and Nick, M. and Tautz, C. :"Concepts for reuse in the experience factory and their implementation for CBR system development". In Proceedings of the Eleventh German Workshop on Machine Learning, August 1998.
3. Bach, V., Vogler, P. Oesterle, H.: "Business Knowledge Management", Springer, 1999.
4. Bartlmae, K.: "An Experience Factory Approach for Data Mining". In Proceedings of the second Workshop: "Data Mining und Data Warehousing als Grundlage Entscheidungsunterstützender Systeme (DMDW99)", Univ. Magdeburg, September 1999.
5. Bartlmae, K.: "A CBR based Experience Factory for Data Mining". In Proceedings of the International Computer Science Conference (ICSC'99): Internet Applications, Lecture Notes of Computer Science, Hong Kong. Springer, Dezember 1999.
6. Bergmann, R., Breen, S., Göker, M., Manago, M., Wess, S.: "Developing Industrial Case-Based Reasoning Applications". Lecture Notes in Artificial Intelligence 1612, Springer; Berlin, Heidelberg, New York, 1999
7. Basili, V. R., Caldiera, G. , Rombach., H. D.: "Experience Factory". In John J. Marciniak, editor, Encyclopedia of Software Engineering, vol.1, pp.528-532. John Wiley and Sons, 1994.

8. Brachman, R.J.; Anand, T.: "The Process of Knowledge Discovery in Databases: A Human-Centered Approach". In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurasamy, R. (eds.). Advances in Knowledge Discovery and Data Mining. AAAI /MIT Press, 1996.
9. Berry, M.J.A., Linhoff, G.: "Data Mining Techniques. For Marketing, Sales and Customer Support". Wiley Computer Publication, 1997
10. Cabena, P., Hadjinian, P., Stadler, R., Verhees, J, Zanasi, A.:"Discovering Data Mining. From Concept To Implementaion". Prentice Hall, Upper Saddle River, New Jeresey 07458, 1998.
11. Cho, J.R., Mathews, R.C.:" Interactions Between Mental Models Used in Categorization and Experiential Knowledge of Specific Cases". The Journal of Experimental Psychology, 49A (3), 1996.
12. CRISP-DM, March 1999, DaimlerChrysler, Forschung und Technologie, 1999, http://www.ncr.dk/CRISP.
13. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: "Form Data Mining to Knowledge Discovery: An Overview". In: U.Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy editors, Advances in Knowledge Discovery and Data Mining. MIT Press, Cambrige, MA, 1995.
14. Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.: "The KDD Process for Extracting Useful Knowledge from Volumes of Data". In Communications of the ACM, November 1996, Vol. 39, No. 11, pp. 27-34, 1996.
15. Gresse von Wangenheim, C., Moraes, A. R., Althoff, K.-D., Barcia, R. M., Weber, R. , Martins, A.: "Case-Based Reasoning Approach to Reuse of Experiential Knowledge in Software Measurement Programs". Proc. of the 6th German Workshop on Case-Based Reasoning, Berlin, Germany, 1998.
16. Hand, D. J.: "Construction and Assessment of Classification Rules". John Wiley and Sons Ltd., Sussex, England, 1997.
17. Heisig.: "Wissensmanagement in Deutschland und Europa - Stand und Entwicklung. Ergebnisse und TOP 200 europ"aischen Unternehmen". In: R. Schmidt (Ed.), 21. Online Tagung der DGI: Aufbruch ins Wissensmanagement, Deutsche Gesellschaft f"ur Informationswissenschaft und Informationspraxis (DGI), 1999.
18. Hipp, Lindner: "Analysing Warranty Claims of Automobiles. An Application Description Following the CRISP-DM Data Mining Process". In Proceedings of the ICSC'99, Springer, December 1999.
19. Houdek, F., Bunse, C.:"Transfering Experience: A Practical Approach and its Application on Software Inspection". IESE-Report No. 008.99/E, Fraunhofer Institut, 1999.
20. Lenz, M., Hbner, A., and Kunze, M.: "Textual CBR" in: Lenz, M., Bartsch-Spörl, B., Burkhardt, H.D., Wess, S. (Eds.): Case Based Reasoning Technology, From Foundations to Applications ". Lecture Notes in Artificial Intelligence, Vol. 1400, Springer, Berlin, Hamburg, 1998.
21. Probst, G., Raub, S., Romhardt, K.: "Wissen Managen", Gabler, 1999.
22. Reinartz, T., and Wirth, R.:"The Need for a Task Model for Knowledge Discovery in Databases". In: Kodratoff, Y., Nakhaeizadeh, G., and Taylor, C. (eds.). Workshop Notes Statistics, Machine Learning, and Knowledge Discovery in Databases. MLNet Familiarization Workshop, April, 28-29, Heraklion, Crete, pp. 19-24, 1995.

Completing missing values using discovered formal concepts

S. Ben Yahia [1], Kh. Arour [1], A. Jaoua [2]
[1]Faculty of Sciences of Tunis
Computer Science Department,
Campus Universitaire, 1060 Tunis, Tunisia.
E-mail: **sadok.benyahia@fst.rnu.tn– mohamed.arour@planet.tn**
[2]King Fahd University for Petroleum and Minerals
Information and Computer Science Department
Dhahran 31261, Saudi Arabia.
E-mail: **ajaoua@ccse.kfupm.edu.sa**

Abstract. Over the past decades there has been a huge increase in the amount of data being stored in databases as well as the number of database applications in business and the scientific domain. This explosion has pointed out the need of techniques or algorithms in order to extract and discover non-trivial, unknown and potentially useful information from large data sets. This extraction of knowledge from large data sets is called Data Mining or Knowledge Discovery in Databases. The extracted knowledge can be used to answer cooperative queries, and facilitate semantic query optimization. Relational databases create new type of problems for knowledge discovery such as missing values for some attributes and a key issue in any discovery system is to ensure the completeness of the discovered knowledge.
In this paper, we address the problem of missing values in relational databases. We present an approach to complete or augment a classical relation containing missing values. This is done by exploiting the useful information yielded by the discovered knowledge represented by formal concepts.
Key words: Relational database, Knowledge discovery, Formal concepts, Galois lattice, Signature, Missing values.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 741–751, 2000.

1 Introduction

The relational data model presenting a solid theoretical background, suffers form an intrinsic insufficiency, due to the blithe assumption that all the information to be represented fits in the relational model. In fact, this assumption can fail in case where the information to be represented is not complete. The easiest way to palliate this insufficiency, is to use a special value, called null value and denoted $\emptyset$, representing the state of lack of complete information. In the literature, a rich body of research has been devoted to the study of the semantic and the different interpretations of null values [2]. In this paper, we are particularly interested in the "no-information" interpretation, in which the attribute is applicable but its actual value, at least for the present moment, is unknown or missing.

In other respects, The last decade has been marked by a rapid growth in our capabilities of both collecting and generating data. Consequently, Knowledge discovery (KD) has become a research topic with an increasing importance [3].

In this paper, we present an approach to complete, or at least augment, a classical relation containing missing value. This is done by exploiting the useful information yielded by the discovered knowledge represented by association rules. This proposed approach can present a notable gain, which can be seen over two facets: i) Complete the instance containing missing values, providing by the way a better description of the real world fragment represented by the database. ii) Following the principle stipulating that "something is better than nothing", we can obtain non empty answers, or at least complete these answers, following a user request involving previously missing data.

In view of this, this paper is organized as follows. In section 2, we present some definitions of the classical relational data model and a brief introduction to the data mining context, Galois connexion and formal concepts. Section 3 introduce the knowledge discovery algorithm based on formal concepts. In section 4, we present an implementation of the algorithm that is based on the signature concept. The complete process of handling missing values is given. Also, the algorithm for completing\augmenting missing values and an illustrative example are given. Section 5 concludes this paper and points out some perspectives of the present work.

2 Formal concepts

2.1 Basic definitions

In this section, we start by presenting some definitions and concepts of relational data model. Then, we present some classical notions of data mining context, Galois connection, formal concepts and formal concept lattice.

Concepts of relational data model: Attributes are symbols taken from a finite set $U = \{A_1, \ldots, A_n\}$ called universe of attributes. Each attribute A_i is associated to a domain denoted $dom(A_i)$. $dom(A_i)$ is the set of possible values

for A_i. Individual attributes are denoted by the initial letters of the alphabet $A, B, \ldots$, and the sets of attributes by the last letters of the alphabet $\ldots, X, Y, Z$. The union of two sets of attributes X and Y is denoted by XY.

A relation r over a relation scheme $R(A_1, ..., A_n)$ is a subset of the Cartesian product $dom(A_1) \times \ldots \times dom(A_n)$. An element of the relation $r, t = (u_1, \ldots, u_n)$, $u_i \in dom(A_i)$, $i = 1, \ldots, n$, is called a tuple. The restriction of t on a set of attributes $X \subseteq U$, denoted by $t.X$, consists of values from t corresponding to the attributes in X.

Definition 1. *A tuple over a set of attributes U is a mapping t that associates with each attribute $A \in U$ either a value of $dom(A)$ or the null value $\emptyset$. The tuple t is said total on A (or A-total) if $t.A$ is not null and is said total if it is A-total $\forall A \in U$ [8].*

Definition 2. *A relation is said total, denoted $r\downarrow$, if $\forall t \in r$, t is total, otherwise it is said partial [8].*

Definition 3. *Let two tuples t and t' defined on the same relation scheme. We say that t subsumes t', written $t \geq t'$, if $t'.A$ is total implies $t'.A = t.A$ $\forall A \in U$. If $t \geq t'$ and t is total, then t is said extension of t', written $t \downarrow \geq t'$ [8].*

Data mining context: A data mining context is a triple $\mathcal{D} = (\mathcal{O}, \mathcal{I}, R)$ describing a finite set $\mathcal{O}$ of objects, a finite set $\mathcal{I}$ of database items and a binary relation R (i.e., $R \subseteq \mathcal{O} \times \mathcal{I}$). Each couple $(o, i) \in R$, means that the object $o \in \mathcal{O}$, has the item $i \in \mathcal{I}$.

Galois connection [5]: Let $\mathcal{D} = (\mathcal{O}, \mathcal{I}, R)$ be a data mining context. For $O \subseteq \mathcal{O}$ and $I \subseteq \mathcal{I}$, we define:

$f(O) : P(\mathcal{O}) \rightarrow P(\mathcal{I})$ $\qquad$ $h(I) : P(\mathcal{I}) \rightarrow P(\mathcal{O})$

$f(O) = \{d \mid \forall g, g \in O \Rightarrow (g, d) \in R\}$ $h(I) = \{g \mid \forall d, d \in I \Rightarrow (g, d) \in R\}$

We can remark that $O \times f(O)$ is the biggest relation of the form $O \times X \subseteq R$, and that $h(I) \times I$ is the biggest relation of the form $I \times X \subseteq R$. In other words, f computes the maximal range for a domain O, and h computes the maximal domain for a range I, as depicted in figure 1.

The operators $h \circ f$ in $\mathcal{O}$ and $f \circ h$ in $\mathcal{I}$ [1], are called Galois connection operators [5].

Given the Galois connection (f, h), then the following properties hold for all $O, O_i, O_j \in \mathcal{O}$ and $I, I_i, I_j \in \mathcal{I}$ [6].

(A1) $O_i \subseteq O_j \Rightarrow f(O_i) \supseteq f(O_j)$ $\qquad$ **(B1)** $I_i \subseteq I_j \Rightarrow h(I_i) \supseteq h(I_j)$

(A2) $O \subseteq h \circ f(O)$ $\qquad$ **(B2)** $I \subseteq f \circ h(I)$

(A3) $f(O) = f \circ h \circ f(O)$ $\qquad$ **(B3)** $h(I) = h \circ f \circ h(I)$

(A4) $O_i \subseteq O_j \Rightarrow h \circ f(O_i) \subseteq h \circ f(O_j)$ $\qquad$ **(B4)** $I_i \subseteq I_j \Rightarrow f \circ h(I_i) \subseteq f \circ h(I_j)$

(A5) $h \circ f(h \circ f(O_i)) = h \circ f(O_i)$ $\qquad$ **(B5)** $f \circ h(f \circ h(I_i)) = f \circ h(I_i)$

(AB6) $O \subseteq f(I) \Leftrightarrow I \subseteq h(I)$

[1] $h \circ f(I) = h(f(I))$ and $f \circ h(O) = f(h(O))$

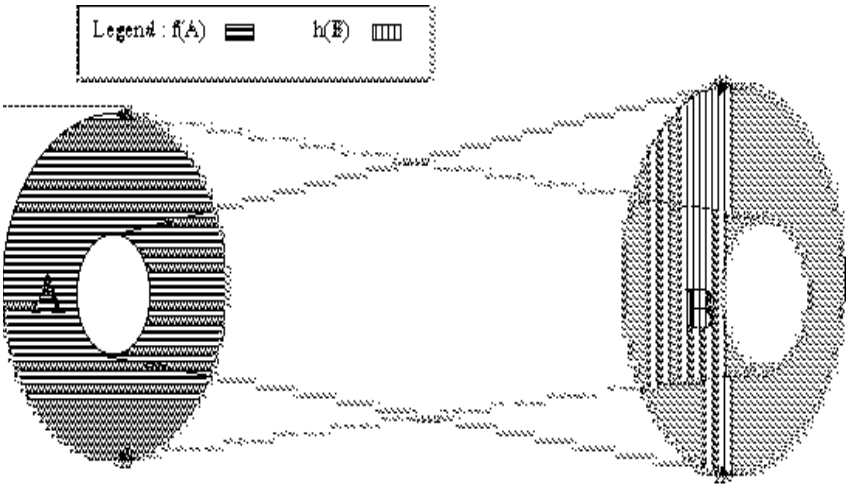


Fig. 1. Galois Connexion

Proposition 1. *Let* $I_1, I_2 \in \mathcal{I}$. *Then* $h\ (I_1 \cup I_2) = h(I_1) \cap h(I_2)$ *[6].*

Proposition 2. $f \circ h(I_1 \cup I_2) = f \circ h(f \circ h(I_1) \cup f \circ h(I_2))$ *[7].*

Formal concept : Let $C \subseteq I$ be a set of items. C is called a *concept*, if and only if it is equal to its *closure*, i.e., $f \circ h(C) = C$. $h(C)$ is called the *domain* of C. Hence, $f \circ h(C)$ is the minimal concept containing C.

Formal concept lattice : Let $\mathcal{C}$ be the set of concepts derived from $\mathcal{D}$ using the Galois connection. The pair $\mathcal{L}_c$=($\mathcal{C}$,<<) is a complete lattice called *formal concept lattice* or Galois lattice.

3 Discovering formal concepts

The pseudo-code for discovering concepts is given in algorithm 1. Notation and parameters used in this algorithm are summarized in table 2. In each iteration, the algorithm constructs a set of candidate concepts (CC), prunes this set, yielding a set of non-redundant concepts. Finally, using this set, it computes the generators set that will be used during next iteration.

Table 2:Notations

CFC_k :	Set of candidate k-itemsets. Each element of this set has three fields: **i)** *gen*: the generator, **ii)** *dom*: the domain and **iii)** *clos*: the closure (i.e., $f \circ h(gen)$).
FC_k :	Set of k-itemsets. Each element of this set has three fields: **i)** *gen;* **ii)** *dom;* and **iii)** *clos*.

Discovering Formal Concepts
Input:$\mathcal{D}$
Output: $FC = \cup_i FC_i$
Begin
$CFC_1 = \{1 - itemsets\}$
For $(i = 1; CFC_i.gen \neq \emptyset; i++)$
do Begin
$CFC_i.clos = \emptyset$
$CFC_i.dom = \emptyset$
$FC_i = Gen\_concepts(CFC_i)$
$FC_i = Gen\_concepts(CFC_i)$
$CFC_{i+1} = Gen\_next(FC_i)$
End
End

Algorithm1:Formal Concepts Discovery.

Function Gen_concepts
Input:CFC_i
Output:FC_i
Begin
$FC_i = \emptyset$
Forall $C \in CFC_i$ **do begin**
$FC_i.gen = C$
$C.dom = h(C)$
$C.clos = f \circ h(C)$
If $h(C) \notin FC_i$ **then**
$FC_i = FC_i \cup \{C\}$
End
End

Function:Gen_concepts.

Hence, initially, CFC_1 is formed by 1-itemsets. Each iteration is composed of two phases:

1. The function *Gen_concept* described below, is applied to each generator in CFC_i, determining its domain and its closure.
2. The set of the generators used in the next iteration, i.e., CFC_{i+1}, is computed by applying the function *Gen_next* described below to FC_i.

The algorithm terminates when no more generators to process, i.e., CFC_i, is empty.

The Gen_concepts function: The function Gen_concepts computes, for all $C \in CFC_i$, the domain and the closure of C. The pseudo-code of Gen_concepts function is given above.

The set CFC_i is pruned off using proposition 3, in order to avoid useless generators computing by removing the redundancy.

Proposition 3. *Let I_1 and I_2 two distinct i-itemsets, such that $f \circ h(I_1) = f \circ h(I_2)$. Then, it is useless to use I_2 as new potential $(i+1)$-itemsets generator.*

Proof. Let $I \in CFC_i$; we have $f \circ h(I \cup I_1) = f \circ h(f \circ h(I) \cup f \circ h(I_1))$. Since $f \circ h(I_1) = f \circ h(I_2)$, then :
$f \circ h(f \circ h(I) \cup f \circ h(I_1)) = f \circ h(f \circ h(I) \cup f \circ h(I_2)) = f \circ h(I \cup I_2)$.

For example, suppose that the set FC_1 contains the 1-itemset generators $\{A, B, E\}$, with respective closures $\{AC\}, \{BE\}, \{BE\}$. The function Gen_next, instead of blindly generating AB, AE, BE as potential 2-itemsets as in Apriori-Gen [1], will remove E from FC_1, since $f \circ h(B) = f \circ h(E)$. Indeed, composing A with B is the same as composing A with E. Hence, using E as a generator will be a source of redundancy.

Gen_next function: The function Gen_next takes as argument the set of concepts FC_i and computes the set CFC_{i+1} containing all $(i+1)$ itemsets, that will be used as generators, during the next iteration.

Gen_next works as follows. We apply the combinatorial phase of Apriori-Gen to the set FC_i. In fact, two distinct generators of size i in FC_i, with the same first $(i-1)$- items are joined, producing a new potential generator of size $(i+1)$.

Insert into $CFC_{i+1}.gen$
Select $p.item_1, p.item_2, \ldots, p.item_i, q.item_i$
From $FC_i.genp, FC_i.genq$
Where $p \neq q, p.item_1 = q.item_1, p.item_2 = q.item_2, \ldots,$
$p.item_{i-1} = q.item_{i-1}, p.item_i < q.item_i$

Finally, a filter, based on proposition 4 and used by [7] is performed. In fact, for each potential generator g in $CLFC_{i+1}$, we test if the closure of one of its i-subsets is a superset of g. In that case, g is removed from $CLFC_{i+1}$.

Proposition 4. *Let I be a generator i-itemset and $S = \{s_1, \ldots, s_j\}$ a set of $(i-1)$-subsets of I where $\cup_{s \in S} s = I$. If $\exists s_a \in S$ such as $I \subseteq f \circ h(s_a)$,then $f \circ h(I) = f \circ h(s_a)$ [7].*

The pseudo code of the third filter is given below.
Forall $p \in LFC_{i+1}.gen$**do begin**
$S_p = \text{Subset}(LFC_i.gen, p)$
// subsets of of p that are existing generators in LFC_i
Foralls $\in S_p$**do begin**
If(p $\subseteq s.clos$)**then**
delete p from $LFC_{i+1}.gen$
End
End

For example, let FC_1=$\{A, B, C\}$, with respective closures $\{AC\}, \{BE\}, \{C\}$. The function Apriori-Gen will generate AB, AC, BC as new potential generators. However, the filter will remove AC , since it is included in $f \circ h(A)$, which implies that computing its closure is useless.

Example 1. Let us consider the transaction database $\mathcal{D}$, given in table 3. Then, figure 2 shows the execution of concepts discovery. The set $CLFC_1$ is initialized with { A, B, C, D, E}. Calling Gen_concept gives for each generator g of $CLFC_1$ its closure. The set LFC_1 is obtained from $CLFC_1$, once all redundant generators are removed (i.e., the generator $\{E\}$ is not considered since $f \circ h(E) = f \circ h(B)$). The set CFC_2 is obtained by applying Gen_next to the set LFC_1. As we can see in figure 2, CFC_2 is equal to $\{AB\}$, $\{BC\}$ and $\{BD\}$. Also, $\{A, C\}, \{A, D\}, \{C, D\}$ of FC_1 do not produce, respectively, the generators $\{AC\}$,$\{AD\}$,$\{CD\}$, since $\{AC\} \subseteq f \circ h(A), \{AD\} \subseteq f \circ h(D), \{CD\} \subseteq f \circ h(D)$. Calling Gen_concepts with CFC_2, gives the domain and the closure of each generator g of CFC_2. After the pruning of CFC_2, we obtain LFC_2 and the

algorithm terminates, since CFC_3 is empty since no generators in FC_2 have the same first 2-item.

Table 3: Transaction DB $\mathcal{D}$

TID	Items
1	ACD
2	BCE
3	$ABCE$
4	BE
5	$ABCE$

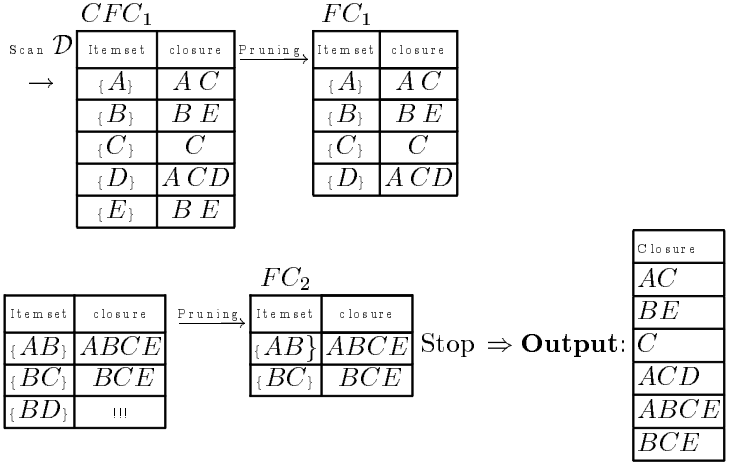


Fig. 2. : Concepts discovery

4 Manipulation of signatures

In this section, we introduce the signature formal concepts (SFC) to calculate $f \circ h$. The core algorithm is based on the generation of closed itemsets, but a bit vector is used to determine the closure of any itemset. Merging of itemsets is easily performed through logical "AND" and "OR" operators applied to a signatures. The key idea of the algorithm is to use a signature to determine which transactions contain which itemsets. To each item, we associate a bit vector (i.e. signature). Bit i is 1 in the signature, if transaction i contains the item, and 0 if not. The requirement is that signatures should be much smaller than objects themselves and should allow closure evaluation without accessing objects (or transactions). The signature (bit vector) acts as a filtering mechanism to reduce the amount of data that needs to be searched for a closure concept. Since the purpose of the signature is to reduce search space in the primary database, an efficient encoding and organization is important. For example, bit-Block compression or Run Length compression can be done to reduce the size of the corresponding signature [4].

4.1 The implemented algorithm

For each basic item (1-itemset) denoted by I, is associated a bit pattern S_I of size n, where n is the number of transactions in DB. Bit i is 1 in this pattern, if transaction i contains this item, and 0 otherwise. The ordered collection of signatures composes the signature file. To calculate any closure for 1-itemsets I, denoted by $S'(I)$, the bit patterns of the associated transactions are AND'ed together.

Example 2. Consider the example transactions database $\mathcal{D}$ given in table 3, the associated signature is presented in table 4. Hence, to calculate $S'(A)$ that represents the closure $f \circ h(A)$, means that we want to discover all items related to A. We can find that by AND'ing the associated columns, where the bits are equal to 1 (i.e. S_1, S_2, S_5). $S'(A) = S'(10101) = S_1 \wedge S_3 \wedge S_5 = 10100$, this means that the items A and C are present together.

Table 4

	Signatures (size =5)				
Items	S_1	S_2	S_3	S_4	S_5
A	1	0	1	0	1
B	0	1	1	1	1
C	1	1	1	0	1
D	1	0	0	0	0
E	0	1	1	1	1

S'				
A	B	C	D	E
1	0	1	0	0
0	1	0	0	1
0	0	1	0	0
1	0	1	0	1

Table 5

	A	B	C	D	E
$S'(A)$	1	0	1	0	1
$S'(B)$	0	1	0	0	1
$S'(C)$	0	0	1	0	0
$S'(D)$	1	0	1	1	0
$S'(E)$	ϕ				
$S'(AB)$	1	1	1	0	1
$S'(AC)$	ϕ				
$S'(AE)$	ϕ				
$S'(BC)$	0	1	1	0	1

The other concepts for 1-itemsets, are given below. The concept of item E is not generated since $S(E) = S(B)$.

Remark 1.
- The closure of E is not generated since $S(E) = S(B)$,
- The closure of AC is not generated since $S(A) \subseteq S(C)$ (all bits that are 1 in $S(A)$ is in $S(B)$), and $S(A)$ is already generated),
- The closure of AE is not generated since $S(A) = S(E)$,
- The closure of BE not generated since $S(B) = S(E)$,
- The same case for the other itemsets (ABE, ABC, $ABCE$).

To implement $f \circ h$, we use signature patterns to generate formal concepts. The implementation is based upon some properties to accelerate the calculus of closures of k-itemsets.

Property 1 *Let $C_1 \subseteq I$ and $C_2 \subseteq I$ be two itemsets from $\mathcal{D}$, if $S(C_1C_2) = S(C_1)$ then $S'(C_1C_2) = S'(C_1)$*

Property 2 *Let $C_1 \subseteq I$ and $C_2 \subseteq I$ be two itemesets from $\mathcal{D}$, if C_1 and C_2 are formal concepts then $S'(C_1C_2) = S'(C_1) \vee S'(C_2)$, where $\vee$ denotes OR operator.*

The bit patterns associated to the concepts of C_1 *and* C_2 *are OR'ed together to form the signature of the concept.* $S'(C_1) = S'_{1C_1}S'_{2C_1}...S'_{mC_1}$, $S'_{iC_1} = 0$ *or* 1 *and* $S'(C_2) = S'_{1C_2}S'_{2C_2}...S'_{mC_2}$,

$$S'(C_1C_2) = S'_{1C_1C_2}S'_{2C_1C_2}...S'_{mC_1C_2} \text{ where, } \begin{matrix} S'_{iC_2C_2} = 1, \text{ if } S'_{iC_2} = 1 \text{ or } S'_{iC_2} = 1 \\ S'_{iC_2C_2} = 0 \text{ otherwise} \end{matrix}$$

Proof. Let C_1 *and* C_2 *be two concepts such that* $C_1 = f \circ h(C_1)$ *and* $C_2 = foh(C_2)$, $foh(C_1 \cup C_2) = foh(foh(C_1)) \cup foh((foh(C_2)) = foh(C_1) \cup foh(C_2) = S'(C_1) \vee S'(C_2)$

4.2 Application

In the following tables, we present a generalized base GBC storing cars characteristics [9].

Table 6: The Sub-base *GBC* without missing values

N	Displace	Fuelcap	Mass	Speed	Cyl	Cost
4	small	low	light	slow	6	cheap
5	large	medium	medium	medium	6	expensive
6	large	medium	light	medium	6	expensive
7	small	low	light	medium	6	cheap
9	medium	low	medium	medium	6	medium
10	medium	high	medium	fast	6	expensive
11	medium	high	light	fast	6	expensive
12	small	high	heavy	medium	4	expensive
14	medium	low	heavy	medium	4	expensive
15	medium	medium	medium	medium	4	medium
17	small	medium	medium	fast	4	medium
18	medium	medium	heavy	slow	4	expensive

Table 7:The Sub-base *GBC* with missing values

N	Displace	Fuelcap	Mass	Speed	Cyl	Cost
1	large	high	medium	medium	$\emptyset$	expensive
2	$\emptyset$	low	$\emptyset$	fast	6	expensive
3	medium	medium	light	fast	$\emptyset$	medium
8	$\emptyset$	medium	light	slow	4	cheap
13	small	$\emptyset$	$\emptyset$	medium	4	cheap
16	$\emptyset$	high	medium	$\emptyset$	4	medium

GBC (Table 6 and table 7) is then replaced by its equivalent signature table, given in table 8, which constitutes the entry of the KD algorithm.

Table 8: The signature table

N	1	2	3							18
DL	1	0		...	...	...	...	...	...	0
DM	0			...	...	...	...	...	...	1
⋮	⋮			⋮	...	...	...	...	...	⋮
CC	0	0		...	...	...	...	...	...	0

Where DL stands for "Displace = *Large*", DM for "Displace = *Medium*", and so on. When we apply the KD algorithm to the above relation, we obtain the following formal concepts: $\{DL, CY6\}$, $\{FH, CY6\}$, $\{MH\}$, $\{DM, CY6\}$,$\{MH, CY4\}$,$\{DM, ML\}$

– **Completing or augmenting partial relation step**

(a) t_1:using **concept1**: t_1**.Cyl**= *6;* $t_1$is completed

(b) t_2:using **concept1**: t_2.**Displace**= *Large;* using **concept3**: t_2.**Mass**= *Heavy;* t_2 is completed
(c) t_3: using **concept5**: t_3.**Cyl**= *6;* t_3 is completed

The augmented partial instance is given hereafter in table 9.

Table 9: The augmented Sub-base GBC

N	Displace	Fuelcap	Mass	Speed	Cyl	Cost
1	large	high	medium	medium	6	expensive
2	large	low	heavy	fast	6	expensive
3	medium	medium	light	fast	6	medium
8	small	medium	light	slow	4	cheap
13	small	$\emptyset$	light	medium	4	cheap
16	medium	high	medium	$\emptyset$	4	medium

Note that the tuples t_{13} and t_{16} are only augmented whereas the remaining tuples are completed.

The gain provided by this approach, can be seen on the following facets:

- Complete or augment a partial instance by providing an estimation of the missing information. Hence, the proposed approach permits to generate and store a value with respect to the discovered rules. This storage of this value, on the one hand, augments the instance and on the other hand, avoids to recompute this value every time it matches a query criteria.
- Provide a non–empty answer to a null query. In fact, let r be a partial instance, Q a query submitted to the system and $Q(r)$ the response generated by the system involving r as a response to Q. In a null query, $Q(r)$ is the empty set. For example, if a user looks for the tuples satisfying the following criteria "Fuelcap = high and Cost = cheap", then the system will generate the empty set as an answer. Using the proposed approach and after the third step, instead of the empty set, the system will find that the tuple #13 satisfies the criteria.

5 Conclusion

Data mining is an emerging research area, whose goal is to extract significant rules from large databases. Many efficient algorithms have been proposed in the literature, e.g., Apriori, Partition, DIC, in the prototypical application of association rule mining, the *market-basket analysis*. They are all based on Apriori mining method, i.e., pruning the itemset lattice, and needing multiple database accesses. In this paper, an algorithm to complete relational relation containing missing values is presented. This algorithm is based upon the useful information contained in the discovered association rules. The KD algorithm is applied in initial relation to discover all formal concepts by using a bit patterns (signatures). The missing values is completed or extended by using the concepts already discovered.

We shall consider the application of the proposed algorithm, to complete missing values in the context of multiple relational tables, connected to each other via key attributes.

References

1. R. Agrawal and R. Skirant. Fast algoritms for mining association rules. In *Proceedings of the 20th Int'l Conference on Very Large Databases*, pages 478–499, June 1994.
2. J. Biskup. *A formal approach to null values in database relations, Vol. 1*, pages 299–341. Plenum Press, 1981.
3. M. Chen, J. Han, and P. Yu. Data Mining: An overview from database perspective. *IEEE Transactions On Knowledge and Data Engineering*, 8(6):866–883, December 1996.
4. C. Faloutsos. Siganture files: Access method for documents and its analytical performance evaluation. In *ACM Transactions on Office Information Systems*, volume 3, pages 237–257, 1998.
5. B. Ganter and R. Wille. *Formal Concept Analysis.* Springer-VErlag, Heidelberg, 1999.
6. A. Jaoua, F. Aloui, S. Elloumi, and S. B. Yahia. Galois connexion in fuzzy binary relation : Applications for discovery of association rules and decision making. In *Proc. of the 5th. Int. Seminar on Relational Methods in Computer Science*, pages 141–149, 9-14 January 2000.
7. N. Pasquier, Y. Bastide, R. Touil, and L. Lakhal. Pruning closed itemset lattices for association rules. In *Proc. of the 14 BDA Conference,*, pages 176–196, December 1998.
8. J. Ullman. *Principles of Database systems.* Computer Science Press, Second Edition, 1982.
9. S. Yen and A. L. Chen. The analysis of relationships in databases for rule derivation. *Journal of Intelligent Information Systems*, 7(3):235–260, 1996.

Extending Datalog with Declarative Updates

Mengchi Liu

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
Email: mliu@cs.uregina.ca

Abstract. The semantics of static deductive databases is well understood based on the work in logic programming. In the past decade, various methods to incorporate update constructs into logic programming and deductive databases have been proposed. However, there is still no consensus about the appropriate treatment of dynamic behavior in deductive databases. In this paper, we propose a language called DatalogU, which is a minimal but powerful extension of Datalog with updates to base relations. DatalogU allows the user to program set-oriented complex database transactions with concurrent, disjunctive and sequential update operations in a simple and direct way. It has a simple and intuitive declarative semantics that naturally accounts for set-oriented updates in deductive databases.

1 Introduction

Deductive databases result from the integration of relational database and logic programming techniques. The semantics of static deductive databases and evaluation of queries are well understood based on the work in logic programming.

Deductive databases differ significantly from logic programming in terms of objectives, semantics and implementation. For example, assume that *emp* is a base relation that gives the monthly salaries and departments of employees. Consider the semantics of the following query:

?– $emp(E, D, S)$

From logic programming point of view, this query requests *one* tuple from the *emp* relation non-deterministically. From deductive databases point of view, this query requests *all* tuples from *emp*. Such a difference also exists between rules of logic programming and deductive databases. Indeed, non-determinism is a fundamental feature of logic programming but it is absent in deductive databases.

Now let us extend the above semantics to updates. Consider the following update queries:

?– $emp(E, D, S), del\ emp(E, D, S)$

It has a normal query $emp(E, D, S)$ followed by a deletion $del\ emp(E, D, S)$. The query finds (marks) the tuples and the deletion deletes the marked tuples. From logic programming point of view, this update query should delete only *one*

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 752–763, 2000.

tuple from *emp* non-deterministically. From deductive databases point of view, this query should delete *all* tuples from *emp*.

In the past decade, various approaches for the inclusion of update capabilities in rule-based languages have been proposed [1, 2, 4, 7, 9, 10, 13, 19, 22, 23, 25, 27, 21]. However, most of them are logic-programming based that support non-deterministic tuple-at-a-time updates and fail to provide a natural account for set-oriented (or bulk) updates in deductive databases syntactically and semantically. Some database-oriented approaches either lack intuitive semantics, such as DatalogA [23] and LDL [24] or expressive power, such as U-Datalog [21]. Surprisingly, most of the proposals, logic-programming based or database-oriented, cannot even support the following simple bulk updates that are expressible in SQL: "give every employee a 10% salary increase" if some employee's salary is zero. The problem is that the updates require the deletion and insertion of the same employee tuple with salary zero, which is disallowed.

Up to now, there is still no consensus about the appropriate treatment of dynamic behavior in deductive databases.

This paper intends to solve the problems. It proposes a language called DatalogU that is a minimal but powerful extension of Datalog with updates to base relations. It allows the user to program set-oriented complex database transactions with concurrent, disjunctive, and sequential update operations in a simple and direct way without the problems that other proposals have. It has a simple and intuitive declarative semantics that naturally accounts for set-oriented updates in deductive databases.

This paper is organized as follows. Section 2 provides an informal overview and examples of DatalogU. Section 3 introduces the syntax of DatalogU. Section 4 defines the semantics of DatalogU. Section 5 discusses the related works. Section 6 presents our conclusions and directions for future research.

2 Informal Overview and Examples

Before introducing the formal syntax and semantics, we present several examples in this section.

Assume that *emp* is a base relation giving the monthly salaries and departments of employees. The *emp* relation can be populated in DatalogU with a number of insertions called transactions as follows:

$?- \; ins \; emp(tom, shoe, 3000)$
$?- \; ins \; emp(ann, shoe, 0)$
$?- \; ins \; emp(sam, toy, 1000)$
$?- \; ins \; emp(pam, toy, 2000)$

Unlike U-Datalog [21], which distinguishes between strong and weak updates, updates in DatalogU are always strong so that if a tuple is already in the relation, it cannot be inserted and if a tuple is not in the relation, it cannot be deleted.

For bulk deletions, DatalogU combines the marking and the deletion into the deletion, so that it is more concise to manipulate a database. Consider the following transactions in DatalogU:

?– $del\ emp(sam, D, S)$
?– $del\ emp(E, toy, S)$
?– $del\ emp(E, shoe, S), ins\ emp(E, toy, S)$
?– $del\ emp(E, D, S), S' = S * 1.1, ins\ emp(E, D, S')$

The first transaction says delete the employee *Sam* from the base relation *emp*. The second says delete every employee in the toy department. The third says transfer all employees from the *Shoe* department to the *Toy* department. The last one says gives all employees a 10% salary increase.

The semantics of DatalogU for the last transaction is as follows. If *EDB* is the current database, then the primitive update $del\ emp(E, D, S)$ finds all bindings for E, D, and S as well as S' and therefore a set DEL of tuples to be deleted can be determined. Based on these bindings, the set INS of tuples to be inserted can be found. As long as the tuples in INS do not occur in $EDB - DEL$, which is the case for this transaction, the updates are performed by first deleting DEL from EDB and then inserting INS to generate the new database EDB', i.e., $EDB' = (EDB - DEL) \cup INS$. Note that if there are employees whose salary is 0 such as $emp(ann, shoe, 0)$, then $emp(ann, shoe, 0) \in DEL$ and $emp(ann, shoe, 0) \in INS$. As $emp(ann, shoe, 0) \notin EDB - DEL$, the updates are still performed. In other approaches such as [9, 21, 27], such updates are inconsistent as there are insertion and deletion of the same tuple and therefore cannot be performed.

Note that there is only one state transition for the above updates even though there is more than one update and updates in DatalogU are set-oriented in the same way as queries in Datalog. Also note that there is only one interpretation for the updates in the above examples that is independent of the order in which *ins* and *del* occur in the command. For this reason, we say that the semantics of DatalogU is declarative.

In fact, using $EDB' = (EDB - DEL) \cup INS$ as the new database is not novel. The idea was first proposed in [22]. However, proper syntax and semantics are not defined. Besides, DatalogU is much more powerful as transactions can be programmed.

If a transaction is to be used more than once, then a generic update rule should be written and stored in the transaction base of the program. For example, the last transaction above can be written as a generic update rule that can be used to give an employee E of the department D a $R\%$ salary increase as follows.

$$raise(E, D, R) \leftarrow del\ emp(E, D, S), S' = S * (1 + R), ins\ emp(E, D, S')$$

In DatalogU, deductive rules and update rules are handled differently. Deductive rules are given the standard static semantics while update rules are not interpreted at all. Update rules are simply treated as named transaction definitions with parameters and are given semantics only when they are invoked by the user, which is similar to DatalogA [23] and LDL [24], but different from U-Datalog [21] in which fixpoint and operational semantics are given to update rules, which have to account for all possible situations, which may never occur.

Consider the following transactions:

$?- raise(tom, shoe, 0.1)$
$?- raise(E, shoe, 0.1)$
$?- raise(E, D, 0.1)$
$?- raise(E, D, R)$

These transactions invoke the transaction definition, i.e., the update rule given above. The first one says give Tom in the shoe department a 10% salary increase. The second says give every employee in the $Shoe$ department a 10% salary increase. The third says give every employee in the database a 10% salary increase. The last one is however problematic, which is shown shortly. In DatalogU, these transactions are expanded using the update rule into the following transactions and are given semantics on their expanded forms.

$?- del\ emp(tom, shoe, S), S' = S * (1 + 0.1), ins\ emp(tom, shoe, S')$
$?- del\ emp(E, shoe, S), S' = S * (1 + 0.1), ins\ emp(E, shoe, S')$
$?- del\ emp(E, D, S), S' = S * (1 + 0.1), ins\ emp(E, D, S')$
$?- del\ emp(E, D, S), S' = S * (1 + R), ins\ emp(E, D, S')$

Since the last one has variables R and S' for which no bindings can be found from the database, it is called not *safe* and cannot be executed.

As update rules are used to extend queries in DatalogU, we disallow recursion by introducing the notion of stratification of update rules.

The above transactions are non-discriminative as employees get the same salary increase. DatalogU also supports discriminative updates called *disjunctive* updates. Consider the following example:

$$?- del\ emp(E, D, S), not\ mgr(D, E), S' = S * 1.1, ins\ emp(E, D, S')$$
$$\&\ del\ emp(E, D, S), mgr(D, E), S' = S * 1.1 + 200, ins\ emp(E, D, S')$$

This transaction says give each employee a 10% salary increase and those in a managerial position an extra \$200. Note that each employee can only get one salary increase, either 10% or 10% + 200, not both. Again, this transaction only causes one database state transition.

The above transaction can be written into two generic update rules in DatalogU as follows.

$$raise(E, D, R) \leftarrow del\ emp(E, D, S), not\ mgr(D, E), S' = S * (1 + R), ins\ emp(E, D, S')$$
$$raise(E, D, R) \leftarrow del\ emp(E, D, S), mgr(D, E), S' = S * (1 + R) + 200, ins\ emp(E, D, S')$$

With these update rules, the above disjunctive transaction can be written as follows:

$?- raise(E, D, 0.1)$

Note that the above disjunctive update rules cannot be expressed directly in many other approaches such as [4, 9, 21, 27]. It is expressible in DatalogA [23] and LDL [24] using the extra *if...then...else* construct.

Note that the above transactions are declarative as the order of *ins* and *del* in them are not relevant. Sometimes, the order of updates may be important

for some transactions. Therefore, DatalogU also supports sequential updates in which the user can specify the order of updates. A sequential transaction has the form of $C_1; ...; C_n$, which specifies that C_i is performed before C_j for $1 \leq i < j \leq n$. The following is an update rule that involves sequential updates. It is adopted from [20].

$$raise\_check(E, D, R) \leftarrow raise(E, D, R); avg\_sal(D, AvgS), AvgS \leq 50K$$

The intention of the rule is to raise the employee salary by $R\%$ only if the average salary in the department stays below 50K after the raise. This rule is not expressible in U-Datalog.

The semantics of sequential updates is given by a sequence of databases. Consider the following sequential transaction:

$$?- raise(E, D, 0.1); avg\_sal(D, AvgS), AvgS \leq 50K$$

The semantics of DatalogU for this transaction is as follows. If EDB is the current database, then $raise(E, D, 0.1)$ produces a new database EDB'. In the new database EDB', the query
$avg\_sal(D, AvgS), AvgS \leq 50K$ is evaluated. If it true, then the whole transaction commits and the database EDB is updated to the new database EDB'. Otherwise, the transaction aborts and the database is left intact.

The next example is adopted from [6, 9].

$$transfer(Acc1, Acc2, Amt) \leftarrow withdraw(Acc1, Amt), deposit(Acc2, Amt)$$
$$withdraw(Acc, Amt) \leftarrow del\ account(Acc, Bal), ins\ account(Acc, Bal'), Bal \geq Amt, Bal' = Bal - Amt$$
$$deposit(Acc, Amt) \leftarrow del\ account(Acc, Bal), ins\ account(Acc, Bal'), Bal' = Bal + Amt$$

However, unlike [6, 9, 27] in which updates are non-deterministic, updates in DatalogU are deterministic and set-oriented. Consider the following example:

$$?- withdraw(A, 50)$$

This query in DatalogU will withdraw \$50 from every account, whereas in [6, 9, 27], \$50 is withdrawn from an account non-deterministically. In order to support set-oriented updates, they provide additional constructs.

3 Syntax of DatalogU

We assume knowledge of the basic concepts related to logic programming, relational and deductive databases [18]. We recall some definitions relevant to our needs and present our notation.

The alphabet of DatalogU consists of a set $\mathcal{C}$ of constants, a set $\mathcal{V}$ of variables, a set $\mathcal{P}$ of predicate symbols partitioned into three disjoint sets: extensional

predicate symbols $\mathcal{P}_{\mathcal{E}}$, intensional predicate symbols $\mathcal{P}_{\mathcal{I}}$, and update predicate symbols $\mathcal{P}_{\mathcal{U}}$.

Terms A variable or a constant is a *term.*

Atoms Let p be a predicate symbol and $T_1, ..., T_n$ terms. Then $p(T_1, ..., T_n)$ is an *atom.* If $p \in \mathcal{P}_{\mathcal{E}}$, then it is an *extensional* atom; if $p \in \mathcal{P}_{\mathcal{I}}$, then it is an *intensional* atom; otherwise, it is an *update* atom. An atom is *ground* if it contains no variables.

Literals A *positive* literal is an extensional or intensional atom. A *negative* literal is a negated extensional or intensional atom.

Expressions An *expression* is defined using $+, -, \times, \div, >, \geq, <, \leq, =, \neq$ in the usual way.

Updates Let A be an extensional atom. Then *ins* A and *del* A are *primitive updates.* An update atom is a *derived update.*

In DatalogU, primitive updates are used to perform set-oriented basic updates to the extensional database, while derived updates must be defined using update rules based on primitive updates or other derived updates and are used to perform complex updates to the extensional database.

Conjunctions A *concurrent conjunction* is of the form $L_1, ..., L_n$, $n \geq 1$ where L_i is either a literal, an expression, or an update. A *disjunctive conjunction* is of the form $N_1 \& ... \& N_n, n \geq 1$ where each N_i is a concurrent conjunction. When $n = 1$, it is called a *trivial* disjunctive conjunction. A *sequential conjunction* is of the form $C_1; ...; C_n$, $n \geq 1$ where each C_i is a concurrent conjunction. When $n = 1$, it is called a *trivial* sequential conjunction.

Our intention is to use a concurrent conjunction to express non-discriminative updates and a disjunctive conjunction to represent discriminative updates that have to be done simultaneously, and a sequential conjunction to represent updates that have to be done in sequence. Therefore, a sequential conjunction cannot appear in a concurrent or disjunctive conjunction.

Limited Terms The limited terms in a concurrent conjunction are defined as follows:

1. a constant is always limited;
2. a term that is in a positive literal or in *del* A is limited;
3. if E_1 and E_2 are limited, so is $E_1 o E_2$ for $o \in \{+, -, \times, \div\}$;
4. if E is limited, so is (E);
5. in $E_1 = E_2$, if E_1 is limited, so is E_2 (and vice versa);

Safe Conjunctions A concurrent conjunction is *safe* if all terms in it are limited. A disjunctive conjunction $N_1 \& ... \& N_n$ is *safe* if each N_i is safe. A sequential conjunction $C_1; ...; C_n$ is *safe* if each C_i is safe.

Deductive Rules A *deductive rule* is of the form $A \leftarrow B$, where the *head* A is an intensional atom and the body B is a concurrent conjunction without updates such that B is safe and all variables occurring in A also occur in B.

Update Rules An *update rule* is of the form $A \leftarrow B$ where the *head* A is an update atom and the body B is a sequential conjunction which contains at least one update such that all variables occurring in A also occur in B.

As an update rule can invoke other update rules, we introduce several syntactic constraints to avoid inconsistency. First, we define the notion of *dependency graph.*

Dependency Graph For a set D of update rules, the *dependency graph* is a graph constructed as follows:

1. the nodes are the update predicates occurring in D
2. there is a marked edge from p to q, denoted by $p \overset{\times}{\rightarrow} q$, if there is an update rule r in which p is in the head and q is in the body which is a non-trivial sequential conjunction
3. there is a non-marked edge from p to q, denoted by $p \rightarrow q$, if there is an update rule r in which p is in the head and q is in the body which is a trivial sequential conjunction

Well-Formed Update Rules A set of update rules is *well-formed* the following hold:

1. its dependency graph has no cycle;
2. there exists no path from p to q such that the incoming edge to q is marked and one edge in the path is not marked in the dependency graph.

The first condition guarantees that the update rule are non-recursive or stratified. The second prevents a sequential conjunction from occurring in a concurrent or disjoint conjunction after expanding.

Consider the following update rules:

$p \leftarrow q, r$
$q \leftarrow s; t$
$r \leftarrow r, u$
...

where p, q, r, s, t, u are update predicates. These rules are not well-formed because there is a circle from r to r and there exists a path $p \rightarrow q \overset{\times}{\rightarrow} s$.

Programs A *program* is a tuple $P = (IDB, TB)$, where IDB is a set of safe deductive rules called *intensional database*, and TB is a set of well-formed update rules called *transaction base.*

Substitutions, bindings and mgus They are defined in the usual way.

Transactions A *transaction* is a sequential conjunction $D_1; ..., D_n$, $n \geq 1$ prefixed with ?–. If $n = 0$, it is a *disjunctive transaction.* Otherwise, it is a *sequential transaction.*

Expanded Transactions Let $P = (IDB, TB)$ be a program and T a transaction. Then we can expand T based on transaction base TB recursively as follows. If A is an update atom in T and $A_1 \leftarrow B_1$, ..., $A_n \leftarrow B_n$ are update rules in P such that A and each A_i can be unified with an mgu θ_i, then replace A with $B_1\theta_1\&...\&B_n\theta_n$ in Q. We denote the expanded update query, which does not contain any derived updates, by $ext(T, TB)$.

Safe Transactions A transaction T is *safe* if $ext(T, TB)$ is safe.

As indicated earlier, we do not give semantics to update rules. Instead, we only give semantics to transactions. Update rules in DatalogU are just used to expand transactions. Therefore, we assume transactions we discuss, are in their expanded form and are safe from now on.

4 Semantics of DatalogU

In this section, we define the semantics of DatalogU transactions.

Herbrand Universe The Herbrand universe U of DatalogU is the set C of constants.

Herbrand Base The *Herbrand base* B is the set of all atoms that can be formed using predicates in $\mathcal{P}_\mathcal{E} \cup \mathcal{P}_\mathcal{I}$ and constants in $\mathcal{C}$. Based on the kind of predicate symbols used, B is partitioned into two subsets: B_E the extensional subset and B_I the intensional subset.

Note that in our semantics we do not interpret update predicates. Instead, they are simply treated as the names of the corresponding updates.

Database A *database* EDB is a subset of B_E. It gives a two-valued interpretation of all base predicates and corresponds to the notion of a database state that is changed by transactions.

Static Semantics For Datalog (with negation), there are several different semantics such as stratified semantics [3, 26], inflational semantics [8], stable model semantics [12], and well-founded semantics [11]. To be general enough, we don't restrict ourselves to any particular semantics. Instead, we introduce the following notion.

Let $P = (IDB, TB)$ be a program and EDB a database. Then $M(EDB, IDB)$ denotes the *intended semantics* of EDB and IDB. Let L be a ground literal. We use $M(EDB, IDB) \models L$ and $M(EDB, IDB) \not\models L$ to denote whether or not L is satisfied by $M(EDB, IDB)$ under this intended semantics.

Given a transaction, we first determine if the queries in the transaction are satisfiable. If so, we can find all the bindings for queries and generate the set of facts to be deleted and the set of facts to be inserted. Then we check if the updates in the transaction are satisfiable by examining the sets to be deleted and inserted. If so, we apply the updates on the database and transit to the new database. If one of the above is not satisfiable, then the transaction cannot be committed and the database is left intact.

4.1 Query Satisfiability

In this subsection, we define the notion of query satisfiability in transactions. Intuitively, it is used to indicate whether the queries in a transaction can be satisfied by a database and a program.

Let $P = (IDB, TB)$ be a program, EDB a database, and T a transaction. Then the notion of *query satisfiability* of T by EDB with respect to P, denoted by $EDB \models_P T$, is defined as follows:

1. For a ground positive literal A, $EDB \models_P A$ if and only if $M(EDB, IDB) \models A$.
2. For a ground negative literal $\neg A$, $EDB \models_P \neg A$ if and only if $M(EDB, IDB) \not\models A$.
3. For a ground expression E, $EDB \models_P E$ if and only if E holds in the traditional interpretation.
4. For a ground primitive update *del* A, $EDB \models_P$ *del* A if and only if $EDB \models_P A$.
5. For a ground primitive update *ins* A, $EDB \models_P$ *ins* A.
6. For a safe concurrent conjunction $C = L_1, ..., L_n$ without derived updates, $EDB \models_P C$ if and only if there exists a ground substitution θ such that $EDB \models_P L_i\theta$ for $1 \leq i \leq n$.
7. For a safe disjunctive conjunction $C = C_1 \& ... \& C_n$ without derived updates, $EDB \models_P C$ if and only if
   (a) there exist a ground substitution θ and a unique i such that $EDB \models_P C_i\theta$;
   (b) there does not exist a binding θ such that $EDB \models_P C_i\theta$ and $EDB \models_P C_j\theta$ for distinct i and j.
8. For a safe concurrent/disjoint conjunction C with derived updates, $EDB \models_P C$ if and only if $EDB \models_P ext(C, TB)$.

Note that the notion of query satisfiability only applies to queries rather than updates. As a deletion in DatalogU implies a query, its satisfiability is only dependent on the query in it. On the other hand, an insertion does not imply a query, so it is always query satisfiable. Note that query satisfiability does not mean that the transaction can be executed.

4.2 Update Satisfiability

In this subsection, we define the notion of update satisfiability in transactions. Intuitively, it is used to indicate whether the updates in a transaction can be satisfied by a database and a program.

Binding Generation Let $P = (IDB, TB)$ be a program, EDB an extensional database, and T a safe transaction such that $EDB \models_P T$. Then the *set of*

bindings for T is
$\Theta = \{\theta \mid \theta$ is a binding for variables in T such that $EDB \models_P T\theta\}$.

Updates Let $P = (IDB, TB)$ be a program, EDB an extensional database, $T = C_1 \& ..., \& C_n$ a safe disjunctive transaction, and Θ the set of bindings for T. Then the *updates* in T is a tuple $U = (DEL, INS)$ where
$DEL = \{A\theta \mid EDB \models_P C_i, del\ A$ is in C_i and $\theta \in \Theta\}$
$INS = \{A\theta \mid EDB \models_P C_i, ins\ A$ is in C_i and $\theta \in \Theta\}$

Proposition 1. Let $P = (IDB, TB)$ be a program, EDB an extensional database, T a safe transaction, and $U = (DEL, INS)$ updates in T. Then $DEL \subseteq EDB$.

Update Satisfiability Let $P = (IDB, TB)$ be a program, EDB an extensional database, T a safe transaction, and $U = (DEL, INS)$ updates in T. Then T is *update satisfiable* by EDB with respect to U, denoted by $EDB \models_U T$, if and only if $INS \cap (EDB - DEL) = \phi$.

Therefore, if a transaction is update satisfiable, then $DEL \subseteq EDB$ and $INS \cap (EDB - DEL) = \phi$. In other words, the facts to be deleted are in the database and the facts to be inserted are not in the database.

4.3 Satisfaction of Transactions

In this subsection, we define the notion of satisfaction of transactions. First, we introduce the following auxiliary notion.

Update Application Let T be a transaction that is update satisfiable by EDB with respect to the updates $U = (DEL, INS)$. Then the *application* of U on EDB, denoted by $\Delta(EDB, U)$, is the database EDB' such that $EDB' = (EDB - DEL) \cup INS$.

Satisfaction of Disjunctive Transactions Let $P = (IDB, TB)$ be a program, EDB an extensional database, T a disjunctive transaction, U updates in T such that $EDB \models_U T$. Then the *satisfaction* of T by EDB and EDB', denoted by $EDB, EDB' \models T$, if and only if $EDB' = \Delta(EDB, U)$

Note that a transaction may have no updates. In this case, DEL and INS are empty sets, $EDB' = EDB$, and the satisfaction of T is reduced to the satisfaction of the queries in T.

Satisfaction of Sequential Transactions Let $P = (IDB, TB)$ be a program, $T = D_1; ...; D_n$ a safe sequential transaction without derived updates, and $EDB_0, ..., EDB_n$ a sequence of databases. Then the *satisfaction* of T by EDB_0, ..., EDB_n, denoted by $EDB_0, ..., EDB_n \models T$, if and only if $EDB_{i-1}, EDB_i \models_{U_i} D_i$ for some updates U_i, $1 \leq i \leq n$.

Theorem 1. Let $P = (IDB, TB)$ be a program, T a sequential transaction, and $EDB_0, ..., EDB_n$ and $EDB'_0, ..., EDB'_m$ are two sequences of databases such that $EDB_0, ..., EDB_n \models T$, $EDB'_0, ..., EDB'_m \models T$. Then $EDB_0 = EDB'_0$ implies $n = m$ and $EDB'_n = EDB'_m$.

Therefore, the sequence of databases associated with a transaction is unique and we can define the semantics of a transaction as follows.

Semantics of Transactions Let $P = (IDB, TB)$ be a program, EDB an extensional database, T a transaction satisfied by a sequence of databases $EDB_0 = EDB$, ..., EDB_n, $n \geq 1$. Then the *semantics* of T is EDB_n.

5 Conclusion

In this paper, we have attempted to formalize the notion of updates in deductive databases. The main contribution is the novel declarative semantics that naturally accounts for set-oriented updates in deductive databases. The update mechanisms described in this paper have been incorporated into the implementations of ROL [14, 15], ROL2 [17] and Relationlog [16].

There are several important issues which we would like to address. Being a database-oriented language, DatalogU does not directly support non-determinism updates that are important in logic programming based approaches [20, 5, 9, 27]. In LDL [24], the choice operator is used to provide non-determinism in deductive databases. We would like to investigate if the explicit use of the existential quantification provides a natural way to support non-determinism in deductive databases. Besides, we intend to extend DatalogU to support rule updates as well. Finally, we like to add aggregates to DatalogU.

References

1. S. Abiteboul. Updates, a New Database Frontier. In *Proceedings of the International Conference on Data Base Theory*, pages 1–18, Pruges, Belgium, 1988. Springer-Verlag LNCS 326.
2. S. Abiteboul and V. Vianu. Datalog Extensions for Database Queries and Updates. *J. Computer and System Sciences*, 43(1):62–124, 1991.
3. K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundation of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Publishers, 1988.
4. A. J. Bonner and M. Kifer. Transaction Logic Programming. In *Proceedings of the International Conference on Logic Programming*, pages 257–279, Budapest, Hungary, 1993. MIT Press.
5. A. J. Bonner and M. Kifer. An Overview of Transaction Logic. *Theoretical Computer Science*, 133(2):205–265, 1994.
6. A.J. Bonner, M. Kifer, and M. Consens. Database Programming in Transaction Logic. In *Proceedings of the International Workshop on Database Programming Languages*, pages 309–337, Manhattan, New York City, 1993. Morgan-Kaufmann.
7. F. Bry. Intensional Updates: Abduction via Deduction. In *Proceedings of the International Conference on Logic Programming*, pages 561–575, Budapest, Hungary, 1990. MIT Press.
8. S. Ceri, G. Gottlob, and T. Tanca. *Logic Programming and Databases.* Springer-Verlag, 1990.

9. W. Chen. Programming with Logical Queries, Bulk Updates and Hypothetical Reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 9(4):587–599, 1997.
10. C. de Maindreville and E. Simon. Modelling Non Deterministic Queries and Updates in Deductive Databases. In *Proceedings of the International Conference on Very Large Data Bases*, pages 395–406, Los Angeles, California, USA, 1988. Morgan Kaufmann Publishers, Inc.
11. A. V. Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of ACM*, 38(3):620–650, 1991.
12. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference and Symposium on Logic Programming*, pages 1070–1080, Washington, USA, 1988. MIT Press.
13. A. C. Kakas and P. Mancarella. Database Updates through Abduction. In *Proceedings of the International Conference on Very Large Data Bases*, pages 650–661, Brisbane, Queensland, Australia, 1990. Morgan Kaufmann Publishers, Inc.
14. M. Liu. ROL: A Deductive Object Base Language. *Information Systems*, 21(5):431 – 457, 1996.
15. M. Liu. An Overview of Rule-based Object Language. *Journal of Intelligent Information Systems*, 10(1):5–29, 1998.
16. M. Liu. Relationlog: A Typed Extension to Datalog with Sets and Tuples. *Journal of Logic Programming*, 36(3):271–299, 1998.
17. M. Liu and M. Guo. ROL2: A Real Deductive Object-Oriented Database Language. In *Proceedings of the 17th International Conference on Conceptual Modeling (ER '98)*, pages 302–315, Singapore, Nov. 16-19 1998. Springer-Verlag LNCS 1507.
18. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2 edition, 1987.
19. S. Manchanda. Declarative Expression of Deductive Database Updates. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 93–100, Philadelphia, Pennsylvania, 1989.
20. S. Manchanda and D. S. Warren. A logic-based language for database updates. In J. Minker, editor, *Foundation of Deductive Databases and Logic Programming*, pages 363–394. Morgan Kaufmann Publishers, 1988.
21. D. Montesi, E. Bertino, and M. Martelli. Transactions and Updates in Deductive Databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):784–797, 1997.
22. L. Naish, L. A. Thom, and K. Ramamohanarao. Concurrent Database Updates in Prolog. In *Proceedings of the International Conference on Logic Programming*, pages 178–189, Melbourne, Victoria, 1987. MIT Press.
23. S. Naqvi and R. Krishnamurthy. Database Updates in Logic Programming. In *Proceedings of the ACM Syum. on Principles of Database Systems*, pages 261–272, Austin, Texas, 1988.
24. S. Naqvi and S. Tsur. *A Logical Language for Data and Knowledge Bases*. Computer Science Press, 1989.
25. R. Reiter. On Specifying Database Updates. *Journal of Logic Programming*, 25(1):53–91, 1995.
26. J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1 & 2. Computer Science Press, 1989.
27. Carl-Alexander Wichert and Burkhard Freitag. Capturing Database Dynamics by Deferred Updates. In *Proceedings of the International Conference on Logic Programming*, Leuven, Belgium, 1997. MIT Press.

Design and Implementation of the OLOG Deductive Object-Oriented Database Management System

Xindong Li and Mengchi Liu

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
Email: {li,mliu}@cs.uregina.ca

Abstract. OLOG is a novel deductive database system for advanced intelligent information system applications. It directly supports effective storage, efficient access and inference of large amount of persistent data with complex structures. It provides a SQL-like data definition language and data manipulation language, and a declarative rule-based query language. It combines the best of the deductive, object-oriented, and object-relational approaches in a uniform framework. This paper describes the design and implementation of the OLOG system.

1 Introduction

Deductive, object-oriented, and object-relational databases are three important extensions of the traditional relational database technology. Deductive databases stem from the integration of logic programming and relational databases. It offers representational and operational uniformity, reasoning capabilities, recursion, declarative querying, efficient secondary storage access, etc. However, deductive databases based on relational databases only allow flat relations and do not support data abstraction. As a result, more powerful deductive languages that support data with complex structures have been proposed, such as LDL [6], LPS [12], COL [1], Hilog [5], Relationlog [16], See [17] for an overview of some of these languages. Also, several deductive database systems that support data with complex structures have been developed, such as LDL [6], CORAL [23], and Relationlog [19].

Object-oriented concepts have evolved in three different disciplines: first in programming languages, then in artificial intelligence, and then in databases since the end of the 60's. Indeed, object orientation offers some of the most promising ways to meet the demands of many advanced database applications. The object-oriented philosophy creates a powerful synergy throughout the development life cycle by combining abstraction, encapsulation, and modularity. In the past decade, various object-oriented data models were developed [3, 7, 8, 13]. Also see an review in [25]. But there is a major problem with the object-oriented approach that is the lack of logical or mathematical foundations that,

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 764–773, 2000.

traditionally, has been playing an important role in database research. Such a foundation is essential for defining the semantics of databases and queries, for database design, and for query optimization.

Object-relational databases combine important object-oriented features with nested relational object databases. It extends relational databases with a richer type system including object orientation and adds constructs to relational query languages, such as SQL to deal with the added data types.

In the past decade, a lot of effort has been made to integrate deductive and object-oriented databases to gain the best of the two approaches, such as O-logic [22], revised O-logic [11], F-logic [10], IQL [2], LOGRES [4], ROL [15], Datalog^{++} [9], and DO2 [14]. As surveyed in [24], many of DOOD models are developed by extending and/or integrating the already existed deductive or object-oriented data models and they either are limited in object-oriented features or lack logical semantics. Few of them are fully implemented as persistent database management systems.

The objective of the OLOG system is to develop techniques for advanced intelligent information systems that directly support effective storage, efficient access and inference of large amount of data with complex structures. The OLOG language [18] is based on IQL [2] and O2 [7]. It overcomes the problem associated with IQL. It effectively integrates useful features in other deductive languages with a well-defined logical semantics.

The OLOG system has been developed in C++ mainly on a SUN SPARC-station running Solaris 2.5. It is implemented as a persistent database system that supports the OLOG query language, an SQL-like data definition language and data manipulation language. The implementation is based on ROL [20] and Relationlog [21].

In this paper, we describe the design and implementation of the OLOG system. In Section 2, we provide a brief overview of the OLOG language. In Section 3, we describe the OLOG system architecture. In Section 4, we explain the file system and storage management. In Section 5, we discuss OLOG kernel. In Section 6, we focus on the query evaluation issues. In Section 7, we conclude and comment on our future plans.

Due to space limitation, the discussion is terse and imcomplete.

2 Overview of OLOG Language

In OLOG, we can have not only classes but also relations with complex data structures such as nested tuples and sets. In this section, we present a brief summary of the OLOG language.

An OLOG database consists of four parts: type, schema, program, and fact. The OLOG language is a typed language for defining, manipulating, and querying OLOG databases.

The type part contains all type definitions. OLOG supports atomic data types including Char, String, Integer, and Real and class types for objects. Based on these data types, complex data types can be defined using tuple and set

constructors. The following examples show how to define tuple and set types in OLOG:

```
create type Address (Street String, City String)

create type PersonSet {Person}
```

The schema part contains all schemas for both classes and relations. The schema provides the description of the database structures and are the basis for storage structures and query optimization strategies. Moreover, they are essential to the consistency of the database. The following examples show how to define classes and relations in OLOG:

```
create class Person (
    Name String,
    Gender Char,
    LivesIn Address,
    Spouse Person,
    Children PersonSet,
    Ancestors PersonSet
    )

create relation Family (
    Husband Person,
    Wife Person,
    Children PersonSet,
    primary key (Husband, Wife)
    )
```

Syntactically, class and relation definitions differ only in the key words. Semantically, a class definition is for a collection of objects whereas a relation definition is for a collection of relation tuples. Objects in OLOG has object identifiers whereas relation tuples do not. OLOG supports class hierarchy and multiple inheritance. The following examples show how to define subclasses in OLOG:

```
create class Employee isa Person (
    Salary Integer,
    Phone Integer
    )

create class Student isa Person (
    Age Integer
    )
```

The rule part is used to define intentional data that can be derived from the extensional data in fact part. Rules correspond to views of relational databases, but can be defined recursively. The following example show how to define rules to derive intensional information for class Person and relation Family:

```
create rule Person (
    Ancestors <P> :- Parents <P>;
    Ancestors <A> :- Parents <P>, P.Ancestors <A>;
    )
create rule Family (
    Family (Husband H, Wife W, Children <C>) :-
        Person H (Gender 'M', Spouse W, Children <C>);
    Family (Husband H, Wife: W, Children <C>) :-
        Person W (Gender 'F', Spouse H, Children <C>);
    )
```

The fact part contains relation tuples and objects. The following example shows how to create objects in OLOG, where tom and pam are object identifiers:

```
insert Employee tom(Name "Tom",
    Gender 'M',
    LiveIn ("Elm St", "Toronto"),
    Spouse pam,
    Parents {}
    )
insert Student pam(Name "Pam",
    Gender 'F',
    LiveIn ("MacPherson Avenue", "Regina"),
    Spouse tom,
    Parents {}
    )
```

The user can use OLOG queries to query OLOG databases. Queries in OLOG are represented using rules as well, the head of which specifies how to construct the result whereas the body specifies the query conditions. The following are several query examples:

```
query Result (TomsWife S) :- Person (Name "Tom", Spouse.Name S)
query Result (TomsAncs <S>) :- Person (Name "Tom", Ancestors.Name S)
query Result (AllTomsChildren <C>) :-
        Family (Husband.Name "Tom"; Children.Name C)
query Result (Person N, Children <C>) :-
        Family (Husband.Name N, Children.Name C)
```

Like ROL and Relationlog, OLOG supports type, schema, rule and other higher-order queries.

3 System Architecture

The OLOG system has been implemented as a single-user persistent database system in C++ under the Unix environment. The functional components of

OLOG can be roughly divided into three layers: interface, OLOG kernel, and storage management as shown in Figure 1.

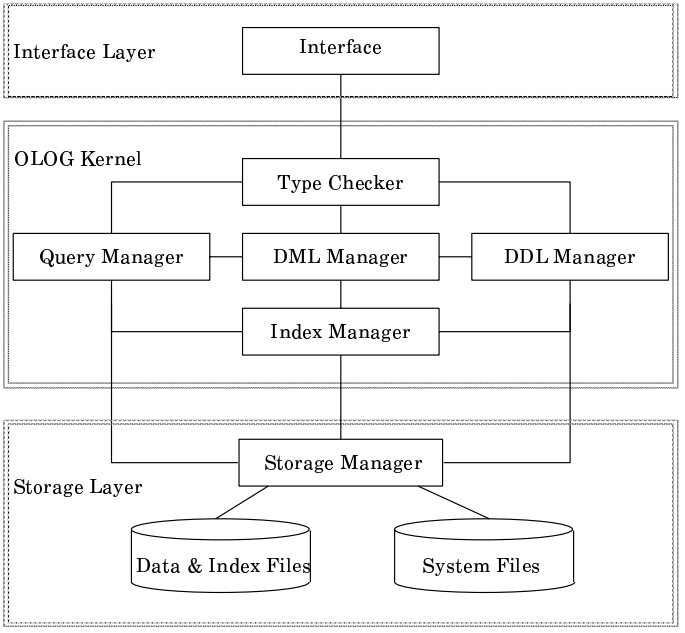


Fig. 1. The Architecture of OLOG System

The interface layer provides a friendly interface to users, takes user requests in different forms, and translates them into internal form to the OLOG kernel. Two kinds of interfaces are provided: textual interface and web interface.

The OLOG kernel consists of five components that are type checker, query manager, DML manager, DDL manager, and index manager. The type checker deals with all the type checking requests. The DDL manager is responsible for processing all DDL commands, maintaining system catalogs about domains, classes, relations and rules. The DML manager performs all the updates to the extensional relations and objects. The query manager handles user queries. The index manager is responsible for maintaining the index information, processing all index related operations from DDL and DML managers, and providing the index information for query manager.

The storage management layer is responsible for the management of the disk-based data structures on which databases are stored. It provides rapid access to objects and relations and meta information about them on the disk. The main job of the storage manager is to move the data between disk and memory as needed.

4 File system and storage management

The OLOG file system is built on top of the UNIX file system. Each database corresponds to two disk files: the data file and the index file, which are automatically created by OLOG system when started. The data file stores objects and relation tuples whereas the index file contains key indices to objects and relation tuples. In addition, OLOG maintains four system files for meta data management: database file which records the information of all databases in the system, relation file which records the information of all relations, class file which records the information of all classes, and rule file which records the information of all rules in each database. All data, index and various definition units in OLOG files are stored as byte string.

The internal structure of the OLOG data file supports the following characteristics:

(1) variable-length relation tuples and objects
(2) indices for objects and relation tuples
(3) reuse of deleted disk file space

The data file contains a file header and a dynamic storage area where variable-length blocks of data are stored. The file header is composed of the following fields:

(1) free space – stores a pointer to the first block of deallocated heap space
(2) end of file – stores a pointer to the address of the last byte in the file
(3) start of heap – stores a pointer to the start of the dynamic storage area

When a block is deallocated, the block is marked deleted and left in the file. The size of the file is determined by blocks allocated and will remain the same no matter how many blocks are deleted. The deleted blocks are maintained in a non-contiguous list. Each block in the list points to the next block in the list starting at a specified address. The "free space" pointer in the file header is used to store the file address of the block where the free space list starts.

The dynamic data area always starts out empty and grows when variable data blocks are allocated. The "free space" and "end of file" pointers in the OLOG file header are used to maintain the dynamic data area. The "end of file" pointer marks the end of the file and points to the location where the file can be extended during block allocation. OLOG always tries to allocate blocks from its free space list.

In OLOG, file addresses of block are used as pointers to the objects and relational tuples. File addresses are represented by 32-bit signed integer values. Therefore, the file can grow to a maximum size of 2.1 GB. File addresses have to be translated into exact memory addresses for the accesses in memory. The Storage Manager is in charge of address mapping that translates file addresses into memory addresses

To make efficient use of each block, OLOG adopts a technique similar to that used in Cache-Memory Mapping, called *Set Associative Mapping*. Each

relation or class has its own buffer space for its own facts. The buffer space is composed of a lot of memory pages, holding some memory slots. This manager is responsible for managing the space occupied by various entities requested by the applications. This means if an object or a relational tuple is currently in main memory, then its memory address is returned to the application requesting it. Otherwise the Storage Manager employs a LRU algorithm to drop some old objects or relational tuples that have not been used for a long time, and then loads this one into memory. In case that the block to be dropped has been changed or deleted, the Buffer Manager physically changes or deletes the block stored on disk. The format of an object in its page buffer is similar to its disk format.

5 OLOG Kernel

In this section, we briefly describe Query, DML, DDL and Index managers in OLOG Kernel.

Query Manager The Query Manager is responsible for data retrieval and rule evaluation pertaining to the facts stored extensionally in the database and defined intensionally by rules. It firstly translates a query into its internal expressions, optimizes them based on the conditions the query processes and then uses different evaluation strategies such as matching, semi-naive bottom-up evaluation with rule ordering, and magic-set rule rewriting techniques to find the results. We discuss query evaluation in detail in Section 6.

DML Manager The DML Manager performs all the updates to the extensional relations and objects. When an object or relation tuple is to be inserted, DML first invokes DDL manager to query the schema definition and check whether it is well-typed with respect to its schema definition. Only well-typed objects and relation tuples can be inserted successfully. When an object or relation tuple is to be deleted, DML first checks whether it is referenced. If not, it then proceeds; otherwise, it rejects the request. Modification of an object or relation tuple is handled similarly. In OLOG, an update may imply a query. In this case, the DML manager will invoke the Query Manager to process the query before performing the update. After extensional relations or objects are updated, it will also request the Query Manager to propagate the updates to the materialized intensional relations and objects that are dependent on the updated ones.

DDL Manager The DDL Manager processes all OLOG DDL commands, maintains system files about domains, class and relation schemas, and rules. It also handles domain, schema, and rule queries and is thus called by DML and Query Managers. As intensional information may be materialized, the DDL manager is also responsible for dropping those materialized intensional information and the corresponding rules when requested.

Index Manager The Index Manager maintains the B-tree indices for the OLOG extensional and intensional relations and objects. and Hash indices for temporal ones. They are chosen because they are efficient and standard, and are used by most database systems. For objects, object identifiers are used as the key. For relations, primary keys are used as the key for B-tree indexing. B-tree indices reside both in memory and on disk and are used for defining and maintaining keys of each relation. They are stored in the index file of each database. Hash indices are applied only in memory for temporal relations, which could be the differential relations of semi-naive evaluation or temporarily created views.

6 Query Evaluation

The OLOG query manager is responsible for data retrieval and rule evaluation pertaining to the facts stored extensionally in the data file and defined intensionally by rules in the rule system file. It combines the query processing methods in ROL [20] and Relationlog [19] and supports the following evaluation strategies:

(1) matching
(2) semi-naive bottom-up evaluation with rule ordering
(3) magic-set rule rewriting technique

Matching In OLOG, the intensional information derived using rules is maintained by the system. It is kept in main memory if the memory space permits and it can be made persistent and maintained current by the system. If a user query concerns only the information in main memory, OLOG can find the query results immediately without any disk operations. If a user query concerns only the information stored on disk, OLOG uses indices if any to find the required information. For example, OLOG will use matching to evaluate the following query:

```
query Result (TomsWife S) :– Person (Name "Tom", Spouse.Name S)
```

Semi-naive bottom-up evaluation This strategy is used to evaluate rules and is the main one for the query processing in deductive database systems. If a user query concerns the intensional information that is not available in main memory or on disk, and the query is so general that no other methods can be used, then OLOG uses this strategy. For example, OLOG will uses semi-naive bottom-up strategy to evaluate the following query:

```
query Result (Person N, Children <C>) :–
      Family (Husband.Name N, Children.Name C)
```

Magic-set rule rewriting technique If a user query contains some constants, then we can only compute intensional information relevant to the query using the magic-set rule rewriting technique as it simulates in semi-naive bottom-up evaluation the pushing of selections that occurs in top-down approaches. Its performance can rival the efficiency of the top-down techniques. For example, OLOG will use magic-set rule rewriting technique to evaluate the following queries:

```
query Result (TomsAncs <S>) :- Person (Name "Tom", Ancestors.Name S)
query Result (AllTomsChildren <C>) :-
      Family (Husband.Name "Tom"; Children.Name C)
```

7 Conclusion

In this paper, we have discussed the design and implementation of the OLOG deductive object-oriented database system. The main novel feature of OLOG is its natural support for persistent classes and relations so that both pure object-oriented and object relational databases can be supported.

A complete implementation as described in this paper has been developed under the UNIX environment. The system will be available over the Internet after further testing, debugging and improving. More information about OLOG system can be found from the web site `http://www.cs.uregina.ca/~mliu/OLOG`.

We are currently extending OLOG into a full-fledged system. We are also developing the interfaces of SQL, OQL, and extended relational algebra and calculus on the top of OLOG in order to make it a useful tool in the database research and teaching.

Acknowledgment

The authors thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for research and equipment grants to M. Liu and the Graduate Studies and Research of the University of Regina for scholarships to X. Li.

References

1. S. Abiteboul and S. Grumbach. COL: A Logic-Based Language for Complex Objects. *ACM Trans. on Database Systems*, 16(1):1–30, 1991.
2. S. Abiteboul and P. C. Kanellakis. Object Identity as a Query Language. *Journal of ACM*, 45(5):798–842, 1998.
3. P. Butterworth, A. Otis, and J. Stein. The Gemstone Object Database Management System. *Communications of the ACM*, 34(10):64–77, 1991.
4. F. Cacace, S. Ceri, S. Crepi-Reghizzi, L. Tanca, and R. Zicari. Integrating Object-Oriented Data Modelling with a Rule-Based Programming Paradigm. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 225–236, 1990.
5. Q. Chen and W. Chu. HILOG: A High-Order Logic Programming Language for Non-1NF Deductive Databases. In W. Kim, J.M. Nicolas, and S. Nishio, editors, *Proceedings of the International Conference on Deductive and Object-Oriented Databases*, pages 431–452, Kyoto, Japan, 1989. North-Holland.
6. D. Chimenti, R. Gamboa, R. Krishnamurthy, S. Naqvi, S. Tsur, and C. Zaniolo. The LDL System Prototype. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):76–90, 1990.

7. O. Deux and others. The Story of O_2. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):91–108, 1990.
8. D. H. Fishman, B. Beech, H. P. Cate, E. C. Chow, T. Connors, J. W. Davis, N. Derrett, C. G. Hoch, W. Kent, P. Lyngbaek, B. Mahbod, M. A. Neimat, T. A. Ryan, and M. C. Shan. Iris: An object-Oriented Database Management System. *ACM Trans. on Office Information Systems*, 5(1):48–69, 1987.
9. H. M. Jamil. Implementing Abstract Objects with Inheritance in Datalog^{neg}. In *Proceedings of the International Conference on Very Large Data Bases*, pages 46–65, Athens, Greece, 1997. Morgan Kaufmann Publishers, Inc.
10. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of ACM*, 42(4):741–843, 1995.
11. M. Kifer and J. Wu. A Logic for Programming with Complex Objects. *J. Computer and System Sciences*, 47(1):77–120, 1993.
12. G. M. Kuper. Logic Programming with Sets. *J. Computer and System Sciences*, 41(1):44–64, 1990.
13. C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The ObjectStore System. *Communications of the ACM*, 34(10):50–63, 1991.
14. T. W. Ling and W. B. T. Lee. DO2: A Deductive Object-Oriented Database System. In *Proceedings of the 9th International Conference on Database and Expert System Applications (DEXA '98)*, pages 50–59, Vienna, Austria, 1998. Springer-Verlag LNCS 1460.
15. M. Liu. ROL: A Deductive Object Base Language. *Information Systems*, 21(5):431 – 457, 1996.
16. M. Liu. Relationlog: A Typed Extension to Datalog with Sets and Tuples. *Journal of Logic Programming*, 36(3):271–299, 1998.
17. M. Liu. Deductive Database Languages: Problems and Solutions. *ACM Computing Surveys*, 30(1):27–62, 1999.
18. M. Liu. OLOG: A Deductive Object Database Language. In *Proceedings of the Workshop on Next Generation Information Technologies and Systems (NGITS '99)*, pages 120–137, Zikhron-Yaakov, Israel, July 5-7 1999. Springer-Verlag LNCS 1649.
19. M. Liu. Query Processing in Relationlog. In *Proceedings of the 10th International Conference on Database and Expert System Applications (DEXA '99)*, pages 342–351, Florence, Italy, August 30-September 3 1999. Springer-Verlag LNCS 1677.
20. M. Liu. The Design and Implementation of the ROL System. *Journal of Intelligent Information System*, 15(2):41–68, 2000.
21. M. Liu and R. Shan. The Design and Implementation of the Relationlog Deductive Database System. In *Proceedings of the 9th International Workshop on Database and Expert System Applications (DEXA Workshop '98)*, pages 856–863, Vienna, Austria, August 24-28 1998. IEEE-CS Press.
22. D. Maier. A logic for objects. Technical Report CS/E-86-012, Oregon Graduate Institute, Beaverton, Oregon, 1986.
23. R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Seshadri. The CORAL Deductive System. *The VLDB Journal*, 3(2):161–210, 1994.
24. P. R. F. Sampaio and N. W. Paton. Deductive Object-Oriented Database Systems: A Survey. In *Proceedings of the 3rd International Workshop on Rules in Database Systems (RIDS '92)*, pages 1–19. Springer-Verlag LNCS 1312, 1997.
25. V. Soloviev. An Overview of Three Commercial Object-Oriented Database Management Systems: ONTOS, ObjectStore, O2. *SIGMOD Record*, 21(1):93–104, 1992.

On Multi-Dimensional Sorting Orders

Walid G. Aref, Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47907
Aref@CS.Purdue.edu

Ibrahim Kamel, Panasonic Information and Networking Technologies Laboratory, Two Research Way, Princeton, New Jersey 08540,
ibrahim@research.Panasonic.com

Abstract. As multimedia applications grow in complexity, more requirements are imposed on software schedulers. It is not always clear that these schedulers are fair to all aspects of the system, or are controllable, in a measurable way, to favor one aspect of the system over the other. The overall goal of this research is to revolutionize the way schedulers are developed. The scheduling problem is formulated as a multi-dimensional sorting problem. Space-filling curves are used to define a linear order for sorting and scheduling objects that lie in the multi-dimensional space. The characteristics of various space-filling curves are studied for sorting and scheduling purposes. Performance measures are proposed to compare the bias, among other characteristics, of a space-filling curve towards any of its dimensions.

1 Introduction

Database and software systems in general are getting more and more complex, calling for an automation of the code generation process. This paper aims towards the automation of one of the crucial parts, mainly the scheduler.

Writing efficient schedulers is becoming a very challenging task, given the increase in demand of such systems. Consider, for example, the case of network-attached storage devices (NASDs) [4, 10] as a building block for a multimedia server (e.g., see [2]). NASDs are "smart" disks that are attached directly to the network. In a multimedia server, a major part of a NASD's function goes towards fulfilling the real-time requests of users. This involves disk and network scheduling with real-time constraints, possibly with additional requirements like request priorities, and quality of service guarantees. Building a scheduler for NASDs is a very challenging and complex task.

Other examples are those of operating systems schedulers (e.g., see [3, 11]) and CPU schedulers (e.g., see [6, 14, 13]). Applications involve multi-threads with different priorities and different real-time constraints. Scheduling the threads in an effective way is a real challenge. We propose a systematic and a scalable way for developing schedulers in such complex scenarios. The general idea is based on modeling the scheduler requests as points in the multidimensional space, where each of the axes represent one of the parameters (e.g., one axis represents the request deadline and the other represents the disk cylinder number). Then the scheduling problem reduces to the problem of finding a linear-order to traverse

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 774–783, 2000.

these multidimensional points. We propose to achieve this by using space-filling curves.

There are several types of space-filling curves (SFCs), e.g., the Peano curve [12], or the Hilbert curve [8]. Figure 1 shows several SFCs for the two-dimensional space. A space-filling curve acts like a thread that passes through every cell element (or pixel) in the n-dimensional space so that every cell is visited only once, i.e., the space-filling curve does not self-intersect. Thus, a space-filling curve imposes a linear order of the cells in the n-dimensional space. Utilizing space-filling curves in scheduling is a novel approach that simplifies the way scheduling algorithms are developed. More importantly, this approach provides a scalable approach to scheduling, regardless of the complexity of the scheduled task or system.

Many studies have been conducted to compare the performance and the spatial features of space-filling orders. The reader is referred to [1, 5, 7, 9]. Up to the authors' knowledge, none of these studies address the scheduling characteristics of space-filling curves.

In this paper, we study the scheduling characteristics of some of the commonly used SFCs, e.g., whether the SFC is biased towards any of the space dimensions over the other dimensions, and whether SFCs may schedule some events in both forward and reverse ordering with respect to any of the dimensions. These as well as other measures are explained in detail in the paper. This will be the milestone for prototyping a core generic scheduler that can be tailored towards scheduling a specific task.

The rest of this paper proceeds as follows. Section 2 introduces the performance measures that we use to study the scheduling quality of the various SFCs. Section 3 gives the results of our empirical study. Section 4 contains conculding remarks and plans for furture work.

2 Modeling of the Sorting/Scheduling Characteristics of Space-Filling Curves

We propose to use the following performance measures to study the scheduling characteristics of SFCs: *Jump*, *Contiguity*, *Reverse-Order*, and *Bias*. In the following section, we formally define each of these performance measures.

2.1 Performance Measure 1: Jumps

Definition: A *jump* in an SFC is said to happen when the distance, along any of the axes, between two consecutive points in the SFC is greater than one.

As an illustration, consider the three-dimensional case. Assume that the two points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ are the coordinate values of two consecutive points p_1 and p_2 in the SFC. We say that an *x-jump* occurs between p_1 and p_2 iff

$$x\text{-}jump : abs(x_1 - x_2) > 1, abs(y_1 - y_2) \leq 1, abs(z_1 - z_2) \leq 1.$$

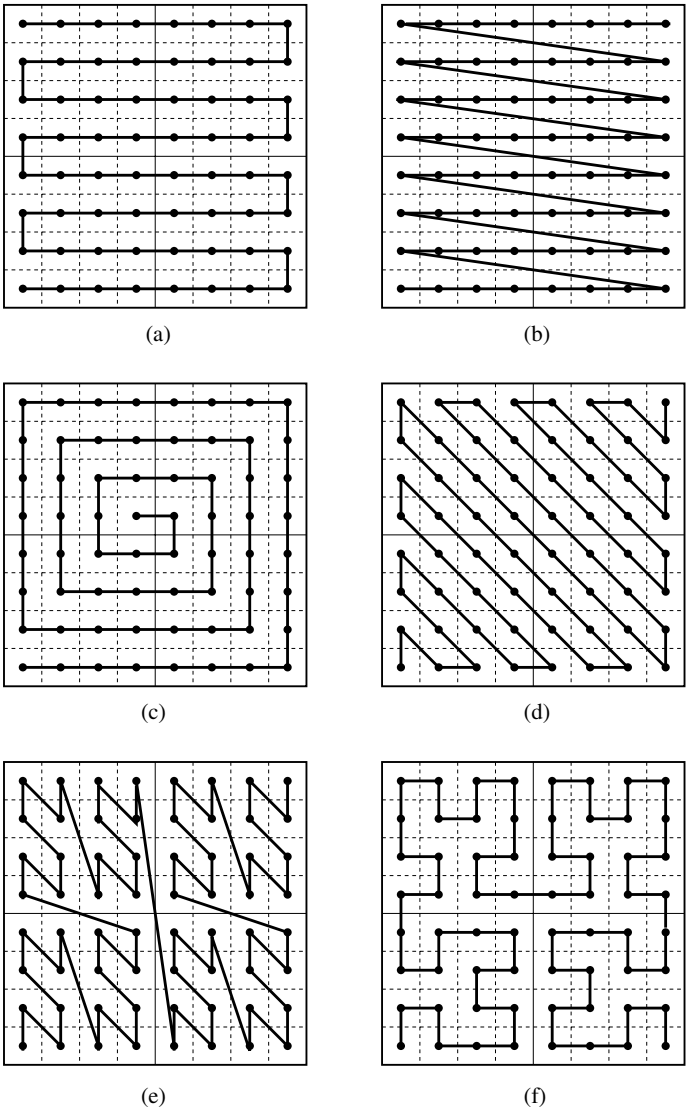


Fig. 1. Various types of space-filling curves in the two dimensional space: (a) circular scan (C-Scan), (b) sweep, (c) spiral, (d) zig-zag, (e) Peano, and (f) Hilbert space filling curves.

Similarly, a *y-jump*/*z-jump* occurs between p_1 and p_2 iff

$$y\text{-}jump : abs(x_1 - x_2) \leq 1, abs(y_1 - y_2) > 1, abs(z_1 - z_2) \leq 1$$
$$z\text{-}jump : abs(x_1 - x_2) \leq 1, abs(y_1 - y_2) \leq 1, abs(z_1 - z_2) > 1$$

A combination of jumps can be defined in a similar way, for example, an *xy-jump* is when an *x-jump* and a *y-jump* occur simultaneously, i.e.,

$$xy\text{-}jump : abs(x_1 - x_2) > 1, abs(y_1 - y_2) > 1, abs(z_1 - z_2) \leq 1$$

Similarly,

$$yz\text{-}jump : abs(x_1 - x_2) \leq 1, abs(y_1 - y_2) > 1, abs(z_1 - z_2) > 1$$
$$xz\text{-}jump : abs(x_1 - x_2) > 1, abs(y_1 - y_2), abs(z_1 - z_2) > 1 \leq 1$$
$$xyz\text{-}jump : abs(x_1 - x_2) > 1, abs(y_1 - y_2) > 1, abs(z_1 - z_2) > 1$$

In addition to these measures, we define an aggregate measure of jumps in an SFC which is the sum of all the above measures, i.e.,

$$JUMP = x\text{-}jump + y\text{-}jump + z\text{-}jump + xy\text{-}jump + yz\text{-}jump + xz\text{-}jump + xyz\text{-}jump$$

Then, we normalize the total number of jumps in relation to the total number of segments in the space, i.e.,

$$JUMP\text{-}rate = JUMP/I,$$

where I is the total segments of cube (N^3).

2.2 Performance Measure 2: Contiguity

Definition: The *contiguity* in an SFC is said to happen when the distance, along any of the axes, between two consecutive points in the SFC is equal to 1.

As an illustration, consider the three-dimensional. Assume that the two points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ are the coordinate values of two consecutive points p_1 and p_2 in the SFC. We say that an *x-contiguity* occurs between p_1 and p_2 iff

$$x\text{-}contiguity : abs(x_1 - x_2) = 1, y_1 = y_2, z_1 = z_2.$$

Similarly, a *y-contiguity*/*z-contiguity* occurs between p_1 and p_2 iff

$$y\text{-}contiguity : abs(y_1 - y_2) = 1, x_1 = x_2, z_1 = z_2$$
$$z\text{-}contiguity : abs(z_1 - z_2) = 1, x_1 = x_2, y_1 = y_2$$

A combination of contiguities can be defined in a similar way, for example, an *xy-contiguity* is when

$$xy\text{-}contiguity : abs(x_1 - x_2) = 1, abs(y_1 - y_2) = 1, z_1 = z_2$$

Similarly,

$$\begin{aligned} yz\text{-}contiguity &: x_1 = x_2, abs(y_1 - y_2) = 1, abs(z_1 - z_2) = 1 \\ xz\text{-}contiguity &: abs(x_1 - x_2) = 1, y_1 = y_2, abs(z_1 - z_2) = 1 \\ xyz\text{-}contiguity &: abs(x_1 - x_2) = 1, abs(y_1 - y_2) = 1, abs(z_1 - z_2) = 1 \end{aligned}$$

In addition to these measures, we define an aggregate measure of contiguity in an SFC which is the sum of all the above measures, i.e.,

$$\begin{aligned} CONTIGUITY = &\ x\text{-}contiguity + y\text{-}contiguity + z\text{-}contiguity \\ &+ xy\text{-}contiguity + yz\text{-}contiguity + xz\text{-}contiguity \\ &+ xyz\text{-}contiguity \end{aligned}$$

Then, we normalize the total number of contiguities in relation to the total number of segments in the space, i.e.,

$$CONTIGUITY\text{-}rate = CONTIGUITY/I,$$

where I is the total segments of cube (N^3).

The Relation Between Jumps and Contiguity Corollary:

$$I = JUMP + CONTIGUITY$$

Proof: will be given in the full paper.
From this formula, we know that:

$$JUMP\text{-}rate + CONTIGUITY\text{-}rate = 1$$

2.3 Performance Measure 3: Reverse Order

Definition: *Reverse Order* in an SFC is said to happen when the projection of any two consecutive points in the SFC on one or more of the axes results in scanning the axis in decreasing order.

As an illustration, consider the three-dimensional case. Assume that the two points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ are the coordinate values of two consecutive points p_1 and p_2 in the SFC. We say that an *x-reverse* occurs between p_1 and p_2 iff

$$x\text{-}reverse : x_1 > x_2, y_2 \geq y_1, z_2 \geq z_1$$

Similarly, a *y-reverse*/*z-reverse* occurs between p_1 and p_2 iff

$$\begin{aligned} y\text{-}reverse &: x_2 \geq x_1, y_1 > y_2, z_2 \geq z_1 \\ z\text{-}reverse &: x_2 \geq x_1, y_2 \geq y_1, z_1 > z_2 \end{aligned}$$

A combination of reverse orders can be defined in a similar way, for example, an *xy-reverse* is when

$$xy\text{-}reverse : x_1 > x_2, y_1 > y_2, z_2 \geq z_1$$

Similarly,

$$yz\text{-}reverse : x_1 > x_2, y_1 > y_2, z_2 \geq z_1$$
$$xz\text{-}reverse : x_1 > x_2, y_2 \geq y_1, z_1 > z_2$$
$$xyz\text{-}reverse : x_1 > x_2, y_1 > y_2, z_1 > z_2$$

In contrast to the reverse order, we can define the *forward* order to be:

$$Forward : x_2 \geq x_1, y_2 \geq y_1, z_2 \geq z_1$$

Hence the following invariant holds:

$$I = x\text{-}reverse + y\text{-}reverse + z\text{-}reverse + xy\text{-}reverse + yz\text{-}reverse + xz\text{-}reverse + xyz\text{-}reverse + Forward$$

2.4 Performance Measure 4: Bias

Bias is one of the most important measures in all the four measures. Bias can be estimated by measuring the number of points on the SFC that need to be traversed before the SFC proceeds a certain amount of steps along one of the axes. For example, consider the two-dimensional sweep SFC. In order to advance from point a three steps in the x-direction, we need to traverse a_x points on the SFC. On the other hand, in order to advance from point a three steps in the y-direction, we need to traverse a_y points on the SFC. If it takes more steps to advance along the y-axis than the x-axis, then we say that the sweep SFC is more biased to the x-axis than the y-axis. Notice that this measure is location-sensitive, i.e., the number of steps needed to advance a number of steps along one of the dimensions depends heavily on the locaion of the starting point on the curve. We propose the following measure for estimating the bias of an SFC.
Definition: Define *X-traverse[k]* to be the average number of points on the SFC that are needed to advance k steps in the x-direction, i.e., from any point (x, y, z) to point $(x + k, y, z)$. We define *Y-traverse[k]* and *Z-traverse[k]* in an analogous way.

A useful measure of goodness is when $k = 1$, i.e., *X-traverse[1]*, *Y-traverse[1]*, and *Z-traverse[1]*. These reflect the average number of points on the SFC that we need to traverse when we advance only one step. We then compute the average and standard deviation of these three values to get $Bias\_Avg[1]$, and $Bias\_Std[1]$ for each SFC.

The measures of goodness $Bias\_Avg[1]$ and $Bias\_Std[1]$ have practical useful meanings. $Bias\_Avg[1]$ reflects the average delay until the SFC advances in a given direction. On the other hand, $Bias\_Std[1]$ reflects the fairness aspects, which is how different it is to advance in one direction in contrast to the other dimensions.

3 Experimental Results

Tables 1–7 give the results of comparing the two- and three-dimensional SFCs with respect to the measures of goodness described in Section 2.

Tables 1 and 2 illustrate how the various SFC differ from each other with respect to the *Jump* measure of goodness. From the tables, the Peano and then the Sweep SFCs exhibit the largest amount of discontinuities by performing the largest percentage of jumps (6.257% and 3.1246%, respectively, for the three-dimensional case, and 12.42% and 3.03%, respectively, for the two-dimensional case). Moreover, the jumps are not homogeneous over all the axes. For example, in the three-dimensional case, the x-axis exhibits the highest ratio of jumps both in the Peano and the Sweep SFCs. All other SFCs show no jumps both in two- and three-dimensional curves, except for the spiral SFC, where it has some slight combined xz-jumps. Notice further that the total percentage of jumps has decreased by half for the Peano curve as we move from the two-dimensional (Total jumps = 12.42%) to the three-dimensional cases (Total jumps = 6.257%). It would be interesting to study the asymptotic behavior of this parameter in the multi-dimensional case.

Tables 3 and 4 give the results of the contiguity measure of goodness. As we have shown from the theorem, given in Section 2.2, contiguity is a complementary measure to jumps. For the three-dimensional case, the spiral and the Hilbert SFCs are the most balanced curves, with respect to contiguity, as they both have almost equal x- y- and z-continguities. The diagonal SFC is biased against the z-axis as it has around 93.9% *xy-contiguity*. The c-scan, sweep, and the Peano SFCs are severely biased towards the x-axis. The two-dimensional case exhibits similar behavior.

Tables 5 and 6 give the results of the reverse-order measure of goodness for the same set of SFCs for the two- and three-dimensional cases, respectively. Using the measure *Total-reverse*, we can deduce that the sweep SFC is the curve with the least amount of reverse-order sorting, while the diagonal SFC has the maximum amount of reverse-order sorting. All the remaining curves have around 50% of the links exhibit sorting in reverse order in one or more of their axes. The spiral and the Hilbert SFCs have equal amounts of reverse-order scheduling per axis.

Table 7 gives the results of the bias experiments for the three-dimensional case. The table shows the measures of goodness: bias-avg[1] and bias-std[1]. From the values of bias-std, we can deduce that the spiral and the diagonal SFCs are not biased at all towards any of their axes. On the other hand, from the values of bias-avg, we can see that these two curves require the longest amount of steps in order to advance in any of their axes. The c-scan and the sweep SFCs represent the worst case in terms of their being biased towards one of their axes over the other axes. The Hilbert and the Peano SFCs represent the best compromise, with the Peano curve being slightly more biased than the Hilbert curve (bias-std) and the Hilbert curve requiring slightly more steps in order to advance along any one of the axes (bias-avg).

	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
x-jump	0%	3.03%	0%	0%	0%	3.13%
y-jump	0%	0%	0%	0%	0%	0%
z-jump	0%	0%	0%	0%	0%	0%
xy-jump	0%	0.0946%	0%	0%	0%	1.79%
yz-jump	0%	0%	0%	0%	0%	0.891%
xz-jump	0%	0%	0.0885%	0%	0%	0.446%
xyz-jump	0%	0%	0%	0%	0%	0%
Total jump	0%	3.1246%	0.0885%	0%	0%	6.257%

Table 1. Summary of the percentage of jumps for a variety of three-dimensional space-filling curves.

	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
x-jump	0%	3.03%	0%	0%	0%	4.11%
y-jump	0%	0%	0%	0%	0%	8.31%
xy-jump	0%	0%	0%	0%	0%	0%
Total jump	0%	3.03%	0%	0%	0%	12.42%

Table 2. Summary of the percentage of jumps for a variety of two-dimensional space-filling curves.

	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
x-contiguity	96.90%	96.90%	34.80%	0.0916%	33.3%	50.0%
y-contiguity	3.03%	0%	33.30%	0.0977%	33.3%	0%
z-contiguity	0.0946%	0%	31.80%	0.0946%	33.4%	0%
xy-contiguity	0%	0%	0%	93.90%	0%	25.0%
yz-contiguity	0%	0%	0%	2.93%	0%	0%
xz-contiguity	0%	0%	0%	2.93%	0%	0%
xyz-contiguity	0%	0%	0.003%	0%	0%	18.8%
Total contiguity	100.0%	96.90%	99.9%	100.0%	100.0%	93.8%

Table 3. Summary of the percentage of contiguities for a variety of three-dimensional space-filling curves.

	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
x-contiguity	96.97%	96.97%	51.52%	2.93%	49.95%	0%
y-contiguity	3.03%	0%	48.48%	3.13%	50.05%	50.05%
xy-contiguity	0%	0%	0%	93.94%	0%	37.54%
Total contiguity	100.0%	96.97%	100.0%	100.0%	100.0%	87.59%

Table 4. Summary of the percentage of contiguities for a variety of two-dimensional space-filling curves.

4 Concluding Remarks

The scheduling and sorting characteristics of various space-filling curves have been studied in this paper. Four scheduling measures of goodness have been proposed. Experiments were conducted to show how each of the space-filling curves varies from the other cuvres with respect to these measures of goodness.

Several issues remain to be studied: (1) the issue of building a generic scheduler based on space-filling curves, (2) the automatic generation of schedulers that satisfy some specified requirements, (3) the development of a scheduler-specification language so that the user may specify the requirements of each dimension of the scheduling problem at hand and the characteristics of each dimension, e.g., whether that dimension would favor from reverse ordering or not, (4) the development of techniques to handle the dynamic aspects of scheduling, and finally (5) how to handle the variation in workloads, and possibly being able to adapt by switching from one SFC to another that is more suitable to the change in the workload.

References

1. D.J. Abel and D.M. Mark. A comparative analysis of some two-dimensional orderings. *International Journal of Geographical Information Systems*, 4(1):21–31, January–March 1990.
2. Cuneyt Akinlar, Walid G. Aref, Ibrahim Kamel, and Sarit Mukherjee. Automatic disks: The building block for a scalable distributed file system. In *Fifth International Workshop on Multimedia Information Systems*, Palm Springs Desert, California, October 1999.
3. L. Alger and J. Lala. Real-time operating system for nuclear power plan computer. In *Proc. Real-time Systems Symposium*, December 1986.
4. Garth A. Gibson et. al. File server scaling with network-attached secure disks. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics '97)*, Seatle, Washington, June 1997.
5. C. Faloutsos. Analytical results on the quadtree decomposition of arbitrary rectangles. *Pattern Recognition Letters*, 13(1):31–40, January 1992.
6. B. Ford and S. Susarla. CPU inheritance scheduling. In *Proc. 2nd. USENIX OSDI*, Oct. 1996.
7. M.F. Goodchild and A.W. Grandfield. Optimizing raster storage: An examination of four alternatives. In *Proceedings of the 6th International Symposium on Automated Cartography*, pages 400–407, Ottawa, Canada, October 1983.
8. D. Hilbert. Ueber stetige abbildung einer linie auf ein flashenstuck. *Mathematische Annalen*, 38:459–460, 1891.
9. H. V. Jagadish. Linear clustering of objects with multiple attributes. In Hector Garcia-Molina and H. V. Jagadish, editors, *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, volume 19, pages 332–342, Atlantic City, NJ, June 1990.
10. R. H. Katz. High-performance network- and channel-attached storage. *Proceeding of the IEEE*, 80(8), August 1992.
11. S. Khanna, M. Sebree, and J. Zolnowsky. Real-time scheduling in SunOS 5.0. In *Proc. Winter USENIX Conference*, San Francisco, CA, January 1992.

12. G. Peano. Sur une courbe qui remplit toute une aire plaine. *Mathematische Annalen*, 36:157–160, 1890.
13. David K.Y. Yau. ARC-H: Uniform CPU scheduling for heterogeneous services. In *IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, 1999.
14. David K.Y. Yau and Simon S. Lam. Adaptive rate-controlled scheduling for multimedia applications. *IEEE/ACM Transactions on Networking*, 5(4), August 1997.

	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
x-reverse	48.4%	3.03%	16.7%	48.4%	16.6%	25.0%
y-reverse	1.51%	0%	16.7%	48.4%	16.7%	0%
z-reverse	0%	0%	15.1%	2.93%	16.6%	0%
xy-reverse	0%	0.0946%	0%	0%	0%	14.3%
yz-reverse	0%	0%	0%	0%	0%	7.14%
xz-reverse	0%	0%	0.0916%	0%	0%	3.57%
xyz-reverse	0%	0%	0%	0%	0%	0%
Total reverse	50.0%	3.12%	48.59%	99.7%	50.0%	50.0%

Table 5. Summary of the percentage of reverse-order sorting for a variety of three-dimensional space-filling curves.

	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
x-reverse	48.48%	3.03%	25.02%	46.92%	23.46%	16.62%
y-reverse	0%	0%	23.46%	47.02%	25.02%	33.33%
xy-reverse	0%	0%	0%	0%	0%	0%
Total reverse	48.48%	3.03%	48.48%	93.94%	48.48%	49.95%

Table 6. Summary of the percentage of reverse-order sorting for a variety of two-dimensional space-filling curves.

Goodness Measure	C-Scan	Sweep	Spiral	Diagonal	Hilbert	Peano
Bias-Avg(1)	1387	1387	2431	2270	1509	1386
Bias-Std(1)	1916	1915	1	0	641	741

Table 7. Summary of the Bias measures of goodness Bias-Avg(1) and Bias-Std(1) for a variety of three-dimensional space-filling curves for a space of size 64X64X64.

Scalable Visual Hierarchy Exploration*

Ionel D. Stroe, Elke A. Rundensteiner and Matthew O. Ward

Worcester Polytechnic Institute, Worcester MA 01609, USA
daniel|rundenst|matt@cs.wpi.edu

Abstract. Modern computer applications, from business decision support to scientific data analysis, utilize visualization techniques to support exploratory activities. However, most existing visual exploration tools do not scale well for large data sets, i.e., the level of cluttering on the screen is typically unacceptable and the performance is poor. To solve the cluttered interface problem, visualization tools have recently been extended to support hierarchical views of the data, with support for focusing and drilling-down using interactive brushes. To solve the scalability problem, we now investigate how best to couple such a near real-time responsive visualization tool with a database management system. This integration must be done carefully, since the direct implementation of the visual user interactions on hierarchical datasets corresponds to recursive query processing and thus is highly inefficient. For this problem, we have developed a tree labeling method, called MinMax tree, that allows the movement of the on-line recursive processing into an off-line precomputation step. Thus at run time, the recursive processing operations translate into linear cost range queries. Secondly, we employ a main memory access strategy to support incremental loading of data into the main memory. The techniques have been incorporated into XmdvTool, a visual exploration tool, to achieve scalability. Lastly, we report experimental results that illustrate the impact of the proposed techniques on the system's overall performance.

Keywords: Hierarchical Structures, Visual Exploration, Recursive Queries, Memory Management, Database Back-end.

1 Introduction

Whether the domain is stock data, scientific data, or the distribution of sales, visualization plays an important role in the analysis. Visualization tools exploit the fact that humans can detect patterns and trends in the underlying data by just looking at it, *without* being aware in advance about what pattern they'll face. Human perception is greatly influenced by the way information is presented. Thus, various techniques for displaying data have been proposed, each of which emphasizes different of its characteristics. However, most of these techniques do not scale well with respect to the size of the data. As a generalization, [7]

* This work is supported under NSF grant IIS-9732897 and NSF CISE Instrumentation grant IRIS 97-29878.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 784–793, 2000.

postulates that any method that displays a single entity per data point invariably results in overlapped elements and a convoluted display that is not suited for the visualization of large datasets.

A new approach has been proposed recently for displaying and visual exploring large datasets [6]. The idea is to present data at different levels of detail based on hierarchical clustering the initial datapoints. The problem of cluttering at the interface level is solved by displaying one level of detail only at a time. However, such hierarchical summarizations result in increasing the size of the input by at least one order of magnitude, making the management of data an issue. While storing the data in main memory and flat files is appropriate for small and moderate sized sets, it becomes unworkable when scaling to large sets.

One solution to this is to integrate visualization tools with back-end DBMSs. The integration cannot be performed *blindly* though. Techniques used for main memory processing are typically not efficient if implemented directly in a database environment. A well known example is sorting. Internal sorting strategies differ significantly from external sorting ones. Another example is presented in this work. The recursive processing involved when navigating through hierarchies in main memory is no longer appropriate when storing those hierarchies on the disk. Instead, we propose using a technique called *MinMax trees* [16] that transforms the recursive processing into a set of fast range queries.

Besides efficiently implementing the visual exploration operations such as zooming and brushing in SQL, we also had to address the problem of caching and managing the results of previous database requests such that subsequent visual operations are as efficient as possible. Furthermore, we applied our proposed solution strategies when coupling the XmdvTool visualization tool [19] with an Oracle DBMS. We also report on results from our performance study of the system.

2 Hierarchical Multivariate Data Visualization

This work was triggered by our goal of adding database support to XmdvTool, a software package (http://davis.wpi.edu/~xmdv) designed for the exploration of multivariate data. The tool provides four distinct visualization techniques (scatterplot matrices, parallel coordinates, glyphs and dimensional stacking) with interactive selections and linked views. Our recent efforts [6, 7] have produced versions of these techniques that allows multi-resolution data presentation.

2.1 Visual Brush-Based Exploration

Selection is a process whereby a subset of entities on a display is isolated for further manipulation, such as highlighting, deleting, or analysis [20]. *Brushing* is the process of interactively painting over a subregion of the data display using a mouse, stylus, or other input device that enables the specification of location attributes [1, 19].

Brushing can be performed in screen or data space to specify a *containment criterion*, i.e., whether a particular point is inside or outside the brush. In *screen space* techniques, a brush is specified by a 2-D contiguous subspace on the screen. In *data space* techniques, a specification consists of either an enumeration of the data elements contained within the brush or the N-D boundaries of a hyper-box that encapsulates the selection.

A third category, namely *structure space* techniques, that allows selection based on structural relationships between data points has recently been introduced in [7]. The *structure* of a data set specifies relationships between data points. This structure may be explicit (e.g., categorical groupings or time-based orderings) or implicit (e.g., resulting from analytic clustering or partitioning algorithms).

A *tree* is one convenient mechanism for organizing large data sets. By recursively partitioning data into related groups and identifying suitable summarizations for each cluster, we can examine the data set methodically at different levels of abstraction, moving down the hierarchy (*drill-down*) when interesting features appear in the summarizations and up the hierarchy (*roll-up*) after sufficient information has been gleaned from a particular subtree.

As described earlier, brushing requires some containment criteria. For our first containment criterion, we augment each node in the hierarchy, that is each cluster, with a monotonic value that controls the level-of-detail. The second is based on the fact that each node in a tree has extents, denoted by the left- and right-most leaf originating from the node. A structure-based brush is thus defined by a subrange of the structure extents and level-of-detail values. Intuitively, if looking at a tree structure from the point-of-view of its root node (Fig. 1), the extent subrange appears as a *focus region* (with the focus point at its center), while the level-of-detail subrange corresponds to a sampling rate factor or a *density*. In a 2-D tree representation, the subranges correspond to a horizontal and vertical selection, respectively (Fig. 2).

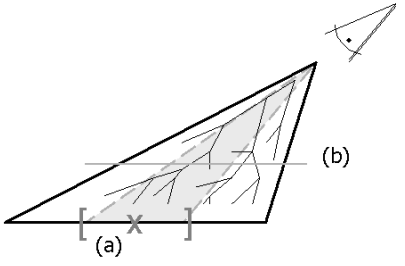


Fig. 1. Structure-based brush: *focus region* (a) and *density factor* (b).

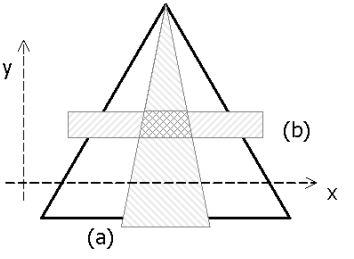


Fig. 2. Structure-based brush: horizontal (a) and vertical (b) selection.

2.2 Brush Semantics

Since a structure-based brush is defined as the intersection of two independent selections, setting such a brush requires two computational phases as well. The first one, the horizontal selection, is accomplished in two steps. In the first step a set of leaf nodes is initially selected based on the order property. Basically, this step corresponds to "select all leaves between the two extreme values e_1 and e_2". In the second phase, the initial selection is propagated up towards the root based on what we term an ALL semantic: "select nodes that have *all* their children already selected" (other semantics like ANY or $MOST$ are also possible [16]). The second computation phase, the vertical selection, consists of refining the set of nodes generated in phase one. Nodes on the desired level-of-detail are only retrieved out of the whole phase one selection.

The brush operations, as described above, are inherently recursive. Recursive processing in relational database systems can be extremely time consuming and thus unsuitable for interactive applications. In Section 3 we develop equivalent but non-recursive computation methods for setting structure-based brushes based on assigning some precomputed values to the nodes that recast retrievals as range queries.

3 MinMax Trees: Translating the Navigation Operations

The question addressed in this section is "how do we translate the visualization operations into database operations". For this purpose we have developed a technique called a *MinMax tree* [16]. The method places the recursive processing in a *precomputation* stage, when labels are assigned to all nodes. The labels provide a containment criterion. Thereafter, simply by looking at a node's label, we are able to determine whether that node belongs to the active selection or not.

3.1 MinMax Tree Definition

A MinMax tree is a n-ary tree in which nodes correspond to open intervals defined over a totally ordered set, called an *initial set*. The leaf nodes in such a tree form a sequence of non-overlapping intervals. The interior nodes are unions of intervals corresponding to their children. The initial set can be continuous (such as an interval of real numbers) or discrete (such as a sequence of integers). In either case, the nodes are labeled as pairs of values: the extents of their interval. As the intervals are unions of child intervals, it follows that a node will be labeled with the minimum extent of its first interval and the maximum extent of its last interval. A node n having two children $c_1 = (\alpha, \beta)$ and $c_2 = (\gamma, \delta)$ will be labeled as $n = (\alpha, \delta)$. Essentially, the process of labeling the nodes is recursive. The intervals are computed and assigned off-line at the time the hierarchy is created and their value and distribution (as well as the tree structure itself) depend on the technique used to create the hierarchy.

Given a MinMax tree T and two nodes x and y of T whose extent values are (x_1, x_2) and (y_1, y_2) respectively, node x is an ancestor of node y if and only if $x_1 \leq y_1$ and $y_2 \leq x_2$. The property is based on the intuition that each node in the tree is included in its parent as an interval.

3.2 Using MinMax Trees to Set Structure-Based Brushes

Having a hierarchy labeled as a MinMax tree, we can implement an ALL structure-based brush as a non-recursive operation based on the property that given the brush values v_{min} and v_{max}, the brush represents the union of all nodes $n = (n_1, n_2)$ whose extents are fully contained in the brush interval (v_{min}, v_{max}), i.e., $(n_1, n_2) \subseteq (v_{min}, v_{max})$ [16]. The horizontal selection defined by an ALL structure-based brush in Fig. 3 and the brush values 0.25 and 0.75 is visually depicted in Fig. 4. The selected nodes are underlined.

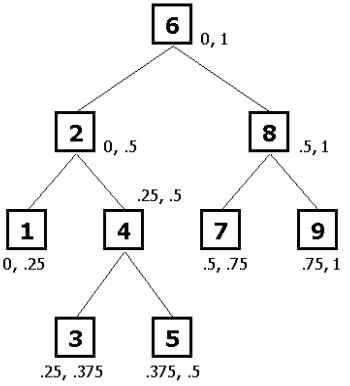


Fig. 3. MinMax tree.

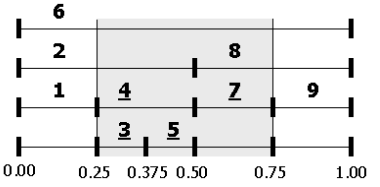


Fig. 4. ALL structure-based horizontal selection.

The value corresponding to the level-of-detail of an aggregation (node) can also be precomputed and stored in the node as part of the label. A vertical selection corresponds further to identifying all nodes at a desired level-of-detail that satisfy the extent selection criterion.

3.3 Translating MinMax Tree Brushes into SQL

The MinMax tree technique reduces the containment criterion from an initially recursive semantic to a simple inclusion test (horizontal selection) and an equality test (vertical selection). We can thus decide whether a node should belong

to the selection or not *independently* from the information stored in the other nodes. One scan of the data is hence sufficient to completely form the selection.

Let H be the relational table that stores the datapoints. Besides the node information, H contains extent and level attributes:

```
H (e_min, e_max, level, ... )
```

An *ALL* structure-based brush for a hierarchy labeled as a MinMax tree and the brush values (min, max, lev) is now a range query, expressed in SQL as:

```
select  *
from    H
where   e_min >= :min
and     e_max <= :max
and     level  = :lev;
```

While a recursive processing would require an exponential processing time, this range query requires only a linear (logarithmic if using indices) processing time for computing brushing results. Tests to confirm the important improvement achieved by our method are presented in Section 5.

4 Memory Management

The question addressed in this section is "how do we organize the data into memory once it arrives from the database". The memory organization is critical in interactive applications since it influences the performance of the subsequent operations. When a new request is issued by the front-end, the difference between the new requested objects and the current content of the buffer has to be quickly computed. Thus, we need to know in each moment what data resides in the memory buffer without fully traversing it.

A characteristic of the objects that are placed in the buffer is that they are not referenced by their IDs when accessed by the front-end. In other words, the front-end doesn't ask for the object x or y. Instead, it passes a query q to the back-end: "are the objects with these characteristics (within this brush) available". Thus, although there are object attributes (the extents) that uniquely identify each entry, a classical lookup by a cache key is not possible when testing whether an object is in the buffer or not. Instead, a query Q is associated with the buffer, similar to *semantic caching* [4]. The two queries q and Q are then compared to determine what objects from q are not in Q, and those objects are retrieved next. The difference results in a new query q_Δ that corresponds to those *to be selected next* objects.

Computing q_Δ is always possible. Since the selected objects are always on the same level it follows that they can be ordered and their extents correspond to disjoint intervals that are consecutive. A selection becomes an interval. Since both q and Q are selections, and thus intervals, and since the difference between two intervals can always be computed, it follows that the difference q_Δ can also be always computed.

The buffer is implemented as a double linked list in which the objects are stored in the increasing order of their extent value. Thus, at any time the buffer content is fully described by the first and the last element in the list, i.e., (e_1, e_2, L) and (e_{n-1}, e_n, L) respectively. An object on level L is already contained in the buffer if its extent interval is included in (e_1, e_n). The look-up uses exactly two buffer accesses.

A significant difference in the buffer management is made by whether the buffer is large enough to store all the objects in the active set or not. Both cases have been analyzed in detail and shown to be efficient and consistent with the buffer access operations in [17].

5 Experimental Results

Experimental Setup. All the experiments were conducted on an Alpha v4.0 878 DEC station, running Oracle 8.1.5. and having no concurrent clients during the tests. We used C as the host language and embedded SQL statements for accessing the the database. The datasets we used had D1=2^{12} (4,096), D2=2^{16} (65,536), D3=2^{17} (131,072) and D4=2^{18} (262,144) tuples in the hierarchical table. All all them represented complete trees with a constant fan-out (2). The number of dimensions was 8.

Experiment 1: MinMax vs. Recursive. First, we measured the system's performance when implementing the structure-based brushes using both the MinMax tree technique and a recursive technique. As SQL does not support recursive views, the recursive technique used an additional attribute to mark the selected tuples along the recursion steps. In both cases we created simple indices on the join attributes and compound indices on extent and level attributes.

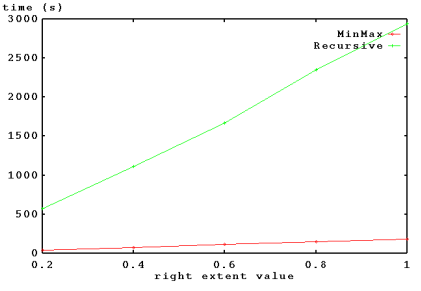


Fig. 5. MinMax vs. Recursive. Structure-based brushes for dataset D3.

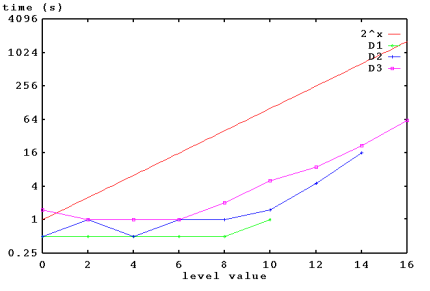


Fig. 6. Varying the level value. Functions compared against 2^x. Logarithmic y scale.

As depicted in Fig. 5), the MinMax method is substantially faster than the recursive one in all cases. These results show clearly that the application of the

MinMax technique in our system made visual exploration over large datasets practically feasible, thus accomplishing our ultimate goal.

Experiment 2: Varying brush parameters. In the next three experiments we analyze the behaviour of the system when the size of the brushes changes. Specifically we vary the level value, the extent values and the size of the dataset. For these experiments we use a compound index on $(e1, e2, L)$ and a *NOCACHE* hint for the Oracle's optimizer (to keep the impact of the Oracle's caching small). Since the brushes are implemented as range queries, we expect that the response time will vary linearly with respect to all these parameters. Indeed, the results show that the time is at most linear with respect to the size of the input (Fig. 6–8).

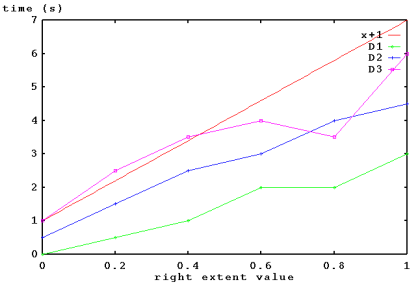


Fig. 7. Varying extent value. Functions compared against $x + 1$.

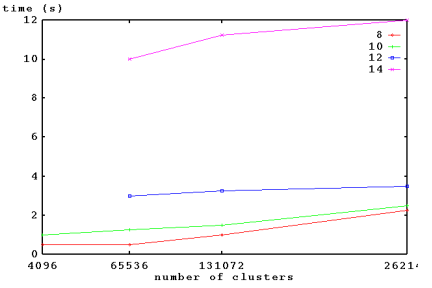


Fig. 8. Varying dataset size. Levels 12 and 14 not defined for D1.

Discussion. The experiments have demonstrated that the processing time for a structure-based brush is only proportional to the number of objects being selected and independent from what brush parameter changed. Thus, the time required for computing a sequence of brushes is proportional to the sum of the objects in all these selections. Due to the incremental computation of the brushes, there is a high probability that small brushes are actually computed rather than large ones. Thus, the amount of the data being read from the database, and implicitly the overall processing, is significantly reduced.

6 Related Work

Visualization-database integration. Integrated visualization-database systems such as Tioga [15], IDEA [14], DEVise [12] represent the work closest related to ours in terms of problem area. The approaches are however different. Tioga [15] implements a multiple browser architecture for what they call a *recipe*, a visual query. However, the problem of query translation is not present since database queries are explicitly specified by the graphical interface. IDEA [14] is

an integrated set of tools to support interactive data analysis and exploration. Some constraints on the data model are imposed by the application domain, but on-line query translation and memory management are not addressed. In DEVise [12], a set of query and visualization primitives to support data analysis is provided. However, caching data is being done at the database level using default mechanisms only.

Other work includes dynamic query interfaces [9], dynamic query histograms [5] and direct manipulation query interfaces [10, 8]. They all have a visual interface and a database back-end. However, the operations translate differently: to dynamic range queries in [5], to temporal queries in [8] and to 2-D spatial queries in [10]. These works do not deal with hierarchy exploration support.

Special techniques for hierarchy encoding. There has been also work in the area of hierarchy representation and querying. Ciaccia et al. [2] used the mathematical properties of *simple continued fractions* for encoding tree hierarchies. However, given a node n, this method cannot efficiently provide the list of descendants of n. This limitation reduces the number of operations that can be supported and makes updates difficult to handle.

Teuhola [18] used a so called *signature* for encoding the ancestor path. Given a node n, the code of n is obtained by applying a hash function to it and by concatenating the resulting value with the code of its parent. The non-unique code can make the quantity of data retrieved be much larger than needed. Moreover, the code obtained by the concatenation of all ancestor codes could exceed the available precision for deep trees.

Caching and buffer management. High level caching in which objects are not individually identified, but rather a set of objects together is identified with the query that generated it is called *semantic caching* [4] or *predicate caching* [11]. Our buffer management requirements are similar to those present in semantic cashing. Other work in the area of object level caching for database applications have been addressed in [3, 13].

7 Conclusions

With the increasing amount of data being accumulated nowadays, the need for visually exploring large datasets becomes imperative. A viable way to achieve scalability in visualization is to integrate visualization applications with database management systems. Such integrations raise two kind of problems: query translation and memory management. This paper presents a solution that addresses both aspects. The approach is being used in coupling XmdvTool, a visualization application for interactive exploration of multivariate data, with an Oracle8i database management system. Experiments for assessing the method showed that, despite the recursive nature of the operations at the interface level, the processing time in our integrated system is only proportional to the number of objects in the active selection. Moreover, the system scales linearly with respect to the size of the dataset.

References

1. A. Becker and S. Cleveland. Brushing scatterplots. *Technometrics, Vol 29(2), p. 127-142*, 1987.
2. P. Ciaccia, D. Maio, and P. Tiberio. A method for hierarchy processing in relational systems. *Information Systems*, 14(2):93–105, 1989.
3. G. P. Copeland, S. Khoshafian, M. G. Smith, and P. Valduriez. Buffering schemes for permanent data. In *Intl Conf on Data Engineering*, pages 214–221, 1986.
4. S. Dar, M. J. Franklin, B. T. Jónsson, D. Srivastava, and M. Tan. Semantic data caching and replacement. In *Intl Conf on Very Large Data Bases*, pages 330–341, 1996.
5. M. Derthick, J. Harrison, A. Moore, and S. Roth. Efficient multi-object dynamic query histograms. *Proc. of Information Visualization*, pages 58–64, Oct. 1999.
6. Y. H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. *IEEE Visualization*, pages 43–50, Oct. 1999.
7. Y. H. Fua, M. O. Ward, and E. A. Rundensteiner. Navigating hierarchies with structure-based brushes. *Information Visualization*, pages 58–64, Oct. 1999.
8. S. Hibino and E. A. Rundensteiner. Processing incremental multidimensional range queries in a direct manipulation visual query. In *Intl Conf on Data Engineering*, pages 458–465, 1998.
9. Y. Ioannidis. Dynamic information visualization. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(4):16–16, Dec. 1996.
10. S. Kaushik and E. A. Rundensteiner. SVIQUEL: A spatial visual query and exploration language. *Lecture Notes in Computer Science*, 1460, 1998.
11. A. M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. *VLDB Journal*, 5(1):35–47, 1996.
12. M. Livny, R. Ramakrishnan, K. S. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and R. K. Wenger. DEVise: Integrated querying and visualization of large datasets. In *ACM SIGMOD Intl Conf on Management of Data*, pages 301–312, 1997.
13. N. Roussopoulos, C.-M. Chen, S. Kelley, A. Delis, and Y. Papakonstantinou. The ADMS project: View R us. *Data Engineering Bulletin*, 18(2):19–28, 1995.
14. P. G. Selfridge, D. Srivastava, and L. O. Wilson. Idea: Interactive data exploration and analysis. In *ACM SIGMOD Intl Conf on Management of Data*, pages 24–34, 1996.
15. M. Stonebraker, J. Chen, N. Nathan, C. Paxson, and J. Wu. Tioga: Providing data management support for scientific visualization applications. In *Intl Conf on Very Large Data Bases*, pages 25–38, 1993.
16. I. D. Stroe, E. A. Rundensteiner, and M. O. Ward. Minmax trees: Efficient relational operation support for hierarchy data exploration. Technical Report TR-99-37, Worcester Polytechnic Institute, Computer Science Department, 1999.
17. I. D. Stroe, E. A. Rundensteiner, and M. O. Ward. Scalable visual hierarchy exploration. Technical Report TR-00-06, Worcester Polytechnic Institute, Computer Science Department, 2000.
18. J. Teuhola. Path signatures: A way to speed up recursion in relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):446–454, June 1996.
19. M. O. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. *Proc. of Visualization*, pages 326–333, 1994.
20. G. Wills. Selection:524,288 ways to say this is interesting. *Proc. of Information Visualization '96, p. 54-9*, 1996.

A Knowledge Based Paradigm for Querying Databases

Paolo Bresciani, Michele Nori, and Nicola Pedot

ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica,
I-38050 Trento-Povo, Italy
{brescian, nori, pedot}@irst.itc.it

Abstract. The so called *Visual Query Systems* try to support unskilled users in formulating complex queries to highly structured databases, by providing graphic paradigms for visualizing the database conceptual models, and by allowing to build queries through visual interactions.
In the present paper we propose a paradigm, for supporting the user in building sophisticated queries, based on a uniform graphical presentation of conceptual models and queries. Its most important novelty is that it is based on a semantic representation of the database conceptual models in terms of Description Logics. In this way it is possible to perform relevant inferential tasks on the query, in order to allow the user to: (i) interactively and iteratively build queries; (ii) be prevented from building inconsistent queries; (iii) interactively explore the database semantics; (iv) be gradually introduced only to those part of the conceptual model relevant for the query formulation; (v) be provided with simple, but effective, features for query refinement and query generalization.

1 Introduction

The task of building a query for accessing data stored in a database can be afforded in different ways by different kinds of user. In any case, the query must be expressed in a form suitable to be processed by the system or the specific application, and that, at the same time, possesses a clear meaning for the user.

Users can be classified on the basis of their familiarity with databases, their knowledge of the application domain, the frequency of use of the application, and the heterogeneity and complexity of the queries they formulate [10, 11]. In particular, due to the increasing availability of complex and highly structured data sources, a growing number of *unskilled* users would like to access data, for several needs, by formulating sophisticated and extemporary queries. These users are likely to have no familiarity with structured query languages, and probably they are just occasional users, unwilling to spend many efforts in trying to understand the way the information is logically organized in the database.

The "Visual Query Systems" (VQS) [11] try to overcome these difficulties by providing graphic paradigms for visualizing the conceptual model of the available data, and by allowing to build queries by means of visual interactions. The VQS are, typically, addressed to all those situations in which: (i) the application

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 794–804, 2000.

domain is complex, and, therefore, a simple form-based, or even QBE like [14], interface is insufficient; (ii) the user is eager to discover interesting information and to educe knowledge from the highly structured set of available data.

Nevertheless, most VQS still fail to adequately support the users in the task of understanding the *semantics* of the data at an intensional level. In fact, although they typically provide graphical representations of the data models, scarcely support the users in strictly linking it with their intuitive understanding of the domain scenario. In order to better understand this connection, the users must perform a set of trials, during which they can *discover* the real meaning of the stored data by analyzing and comparing the answers to different tentative queries. Instead, here we propose a more direct *discovery process*, during which query validation and comparison can be carried out at the intensional level.

Moreover, even assuming a perfect understanding of the conceptual model, its complexity —specially the presence of possible interfering constraints— can lead to an objective difficulty in avoiding inconsistent requests or in noticing similarities, or even equivalences, between apparently different queries. This can lead to two kinds of problem:

1. in the trys of iteratively building the intended query, the user can fail several times, because reiterating slightly different, but all inconsistent, queries;
2. it is possible that, after having submitted a query to a database, the answer is unexpectedly too large or too narrow: in this case, trying to modify the query in a structural and relevant way could be an hard task.

In the present paper we propose a novel paradigm for supporting the user in building sophisticated queries in the context of highly structured object oriented databases. The proposed paradigm is relevant when: (i) the application domain is intrinsically complex; (ii) a hierarchical conceptual model of the domain is available; (iii) the queries the user wants to build are structurally complex; (iv) the user is eager to discover, for best formulating the intended query, the structure of the conceptual model, through an interactive and iterative process.

The proposed paradigm uses, in order to convey the conceptual model and query contents and meanings to the user, a uniform graphical presentation. But the most important characteristic of the paradigm is that it is based on a semantic representation of the database conceptual model, coded into a knowledge base. Thus, it is possible to perform some relevant inferential tasks on the semantics of the queries. In particular, it is possible to allow the user to: (i) interactively and iteratively build queries; (ii) be prevented from building inconsistent queries; (iii) interactively explore the semantics of the queries and of the classes involved, using the *classification* of the queries in the taxonomical conceptual model; (iv) be gradually introduced only to those parts of the conceptual model that are relevant for the query formulation; (v) be provided with simple, but effective, features for query refinement and query generalization.

In section 2, our interaction paradigm will be introduced. To support the paradigm, some reasoning services are needed. They will be discussed in section 3. In section 4, some notes about a first prototypical realization of the paradigm will be given, and section 5 will draw some conclusions.

2 The Interaction Paradigm

In general, submitting a query to a database involves three main steps: (i) build join paths through a set of relational dependencies: in order to do that with traditional systems, the user needs to know details not only on the application domain, but also on the implementation schema; (ii) define arithmetic constraints on the attributes, and select the attributes to be included in the answers; (iii) possibly define some aggregations, and decide a printout format. After these steps, the query can be submitted, and results printed out or shown otherwise.

In the paradigm here presented only the first of these steps is considered. During it, as said above, it is usually assumed that the user already has some knowledge about the application domain and the implementation schema. Instead, in our paradigm, the user is assisted in *understanding* the semantics of the model and the queries iteratively built through successive modifications and refinements. To this end, it is necessary to provide a classification of the queries w.r.t. the set of classes that form the database conceptual model, providing, thus, a way for gradually understanding and learning the semantics of the model. Moreover, classifying the query at each step of the interaction allows to immediately detect semantic inconsistencies w.r.t. the database conceptual model.

2.1 The Approach

In our paradigm a query intuitively corresponds to an object description, with nested relationships with other queries. Therefore, such kind of descriptions should be dynamically generated and managed. From the graphical point of view, a query can be represented as in fig.1, that corresponds to the following:

$$Q(x) \leftarrow \texttt{Prof}(x) \wedge \texttt{hold}(x, y) \wedge \texttt{Lecture}(y) \wedge \texttt{attend}(z, y) \wedge \texttt{CS-stud}(z) \wedge \texttt{hold}(x, w) \wedge \texttt{CS-lect}(w).$$

We believe that the graphical notation of fig.1 is quite easily grasped by the unskilled user, because it is quite close to a natural language *Noun Phrase* (NP), but, at the same time, avoids referential ambiguities. Moreover, the representation shows only the necessary terms of the class hierarchy.

It can be noticed that the query has only one free variable. This is not a real limitation: the selection of other variables can be postponed to the second phase of the list defined before, but this is out of the scope of the present paper.

In our approach, the user is allowed to interactively build sophisticated queries, like that shown in fig.1, by choosing among the classes defined in the conceptual model, and iteratively forming, with them, complex boolean expressions of literals connected by links representing relationships. Of course, it must be possible also to delete or replace parts of the query. In subsection 2.2 the main functionalities for building and modifying queries will be presented. An important aspect is that, during the interaction, the query that is going to be built is constantly guaranteed to be consistent, w.r.t. the conceptual model of the database, in order to avoid the problem 1 listed in section 1. Moreover, in our approach, the query is classified, w.r.t. the conceptual model, after each

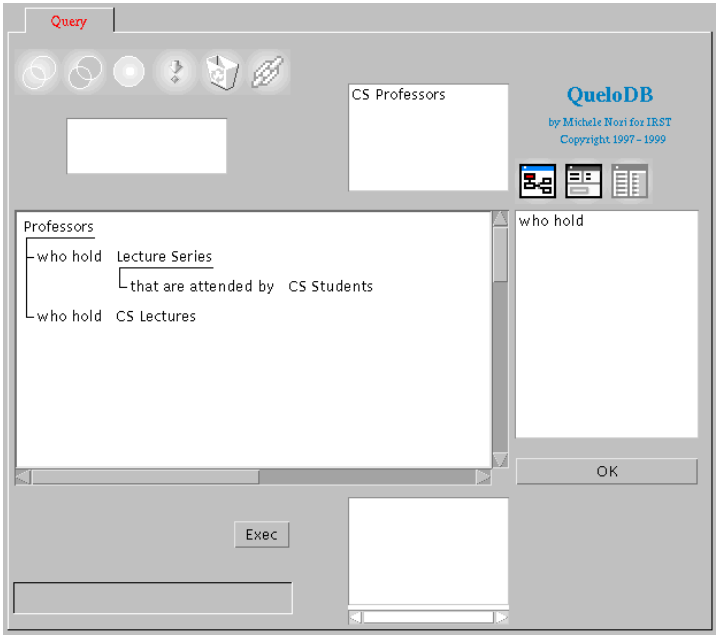


Fig. 1. A possible interface for visually formulating queries.

modification, so that the user can be informed about the position of the query in the conceptual taxonomy. In this way it is possible to face also the problem 2.

Requiring that only consistent queries are built prevents the user to uselessly query the database. Indeed, in order to better achieve this goal, we require some more constraints on the kind of interactions allowed for building the query:

- Only *consistent* actions (i.e., actions that lead to a new query that still is consistent) are allowed.
- Only *relevant* modifications (i.e., query transformations that lead to new queries that are semantically not equivalent to the original one) are proposed.
- Only *close* modifications (i.e., not all the consistent and relevant modifications, but only those that lead to a new query with semantics close to the original query semantics) are proposed. This constraint is required in order to avoid to overload the user with too many options.[1]

In the following we shortly call these constraints "CRCP". A first consequence of CRCP is that, at the beginning of the interaction, the user is allowed to choose among the top classes of the conceptual model hierarchy only.

[1] This point is one of the most delicate. In fact, it is difficult, in general, to give a clear notion of *close* query and *close* modification. Moreover, in some cases, some overload could be acceptable, specially if it give the advantage of shortening the interaction. Anyway, *closeness* must not preclude reachability of every term in the taxonomy.

In the next subsection the principal kinds of interactions will be listed. In order to satisfy CRCP some reasoning tasks have to be performed. They will be dealt with in section 3.

2.2 The Detailed Functionalities

The graphical interface implied by our paradigm includes the following elements: (i) an area where to represent the query in the form shown in fig.1; (ii) a list of names of classes more generic than the query (let's call it a *generalization list*); (iii) a list of names of classes more specific than the query (*specialization list*); (iv) a list of names of classes equivalent to the query (*equivalence list*); (v) a set of buttons or menu that allow to perform various actions for building the query.

Moreover, the list of relationships that can be used to build or modify the query –or a portion of it– must be shown on demand.

As mentioned, the first step of the interaction must be the selection of a starting term among the top classes of the conceptual model, that are shown in the specialization list. After that, the user must be able to modify iteratively the query through the following functionalities:

- *Query replacement.* The goal is to allow the initial navigation in the taxonomy, and the substitution of the whole query with an atomic term semantically close to it, by selecting a replacement among the generalization, the specialization, or the equivalence lists.
- *Relation selection.* It must be possible to restrict the query by imposing the existence of relationships with other atomic terms or complex expressions. A list of relations compatible with the present query must be shown, including, in agreement with CRCP, only those relations that, when used to restrict the query, would lead to a new query that still is consistent. Of course, also the *coherent* and most specific relation range must be automatically proposed.
- *Propositional combination with other terms.* It must be possible to combine selected parts of the query with other terms, to form boolean subexpressions. Also in this case the proposed terms must obey to CRCP, with respect to the kind of boolean combination required.
- *Negation.* Literals should be allowed to be negated,[2] but only if the resulting query will still be consistent the transformation must be allowed.
- *Refinement.* The goal, in this case, is to replace selected subexpressions of the query, instead of combining them using boolean operators. It must be possible to select any kind of subexpression (i.e., also including relationship links). Both generalization and specialization lists for the selected parts must be presented, in order to be selected. Of course, also in this case, it is required that only those generalizations and specializations that satisfy CRCP are shown.

A sample interaction with a first prototypical implementation of the paradigm here presented can be found in [6].

[2] The possibility of negating non-literal expressions is prevented, in order to avoid misleading behaviors and meanings of the boolean combinations described above.

Construct	FOL Semantics
$(\neg C)$	$\neg F_C(\gamma)$
$(C \sqcap D)$	$F_C(\gamma) \wedge F_D(\gamma)$
$(C \sqcup D)$	$F_C(\gamma) \vee F_D(\gamma)$
$(\forall R.C)$	$\forall x.F_R(\gamma, x) \Rightarrow F_C(x)$
$(\exists R.C)$	$\exists x.F_R(\gamma, x) \wedge F_C(x)$
$\vdots$	$\vdots$
(R^{-1})	$F_R(\beta, \alpha)$
$(R \circ Q)$	$\exists x.F_R(\alpha, x) \wedge F_Q(x, \beta)$
$\vdots$	$\vdots$

Table 1. FOL transformational semantics for some DL operators.

3 Reasoning Services

Let's recall that the main requirement of our paradigm is that each functionality that allow to modify a query must obey CRCP. To obtain this behavior some reasoning capabilities are needed. It is here proposed to obtain these capabilities by representing the conceptual model of the database and the query itself in terms of a Description Logic Knowledge Base. In fact, it has been show that Description Logics offer powerful formalisms for solving several problems [4, 3] concerning data modeling [2, 9] and access [2, 5], like, for example, the definition of rich schema languages [8], the schema integration, the inter-view relationship managing, and, in particular, the intelligent query management [7, 3].

In the following we briefly recall some basic notions about Description Logics (DL), and then we exemplify how conceptual models and queries can be represented in terms of DL. Then, it will be possible to introduce the reasoning services that allow to guarantee CRCP for the functionalities listed in subsection 2.2.

3.1 Description Logics

Description Logics are, essentially, variable-free concise reformulations of decidable restricted fragments of First Order Logic (FOL). The syntax of DL allows to express *concepts* (unary predicate symbols), *roles* (binary predicate symbols), and *individuals*[3] (constants). In different DL slightly different languages are used, with different levels of expressiveness. In all of them [12] concept expressions and role expressions can be built starting from the atomic concepts and roles, using different sets of operators, like the concept-forming operators: $\sqcap, \sqcup, \neg, \forall, \exists$, and the role-forming operators: $\cdot^{-1}, \circ$.

Description logics semantics can be given, for example, by mapping DL expressions into FOL formulæ [7]: an atomic concept `A` and an atomic role `P` are

[3] Reasoning about individuals (the *assertional reasoning*) is not relevant for our purposes; therefore, only the *terminological* aspects of DL will be considered.

```
(TOP-DISJOINT: (Prof Lect Stud))

(DEFCLASS Prof
 RELATIONS: ((hold Lect (0 inf)) ...))

(DEFCLASS CS-Prof       ISA: (Prof)
 RELATIONS: ((hold CS-Lect (0 inf))) ...)

(DEFCLASS Eng-Prof      ISA: (Prof)
 RELATIONS: ((hold Eng-Lect (0 inf))) ...)

(DEFCLASS Lect
 DISJOINT: ((Seminar Course))
 DISJOINT-COVERING: ((CS-Lect Eng-Lect))
 INVERSE: ((hold Prof (1 1))) ...)

(DEFCLASS CS-Lect       ISA: (Lect)
 INVERSE: ((hold CS-Prof (1 1)) ...)

(DEFCLASS Eng-Lect      ISA: (Lect)
 INVERSE: ((hold Eng-Prof (1 1)) ...)

(DEFCLASS Seminar       ISA: (Lect)
 INVERSE: ((participate Stud (0 inf))) ...)

(DEFCLASS Course      ISA: (Lect)
 INVERSE: ((attend Stud (0 inf))) ...)

(DEFCLASS Stud
 DISJOINT: ((CS-Stud Eng-Stud))
 RELATIONS: ((participate Seminar (0 inf))
             (attend Course (1 inf))) ...)

(DEFCLASS CS-Stud     ISA: (Stud)
 RELATIONS: ((attend CS-Course (1 inf))) ...)

(DEFCLASS Eng-Stud    ISA: (Stud)
 RELATIONS: ((attend Eng-Course (1 inf))) ...)

(DEFCLASS CS-Seminar  ISA: (CS-Lect Seminar))

(DEFCLASS CS-Course   ISA: (CS-Lect Course)
 INVERSE: ((attend CS-Stud (0 inf))))

(DEFCLASS Eng-Seminar ISA: (Eng-Lect Seminar))

(DEFCLASS Eng-Course ISA: (Eng-Lect Course)
 INVERSE: ((attend Eng-Stud (0 inf))))
```

Fig. 2. Conceptual model.

mapped –or *interpreted*– into the FOL open atomic formulæ $A(\gamma)$ and $P(\alpha, \beta)$. In table 1, for some DL concept expressions C, D the corresponding FOL open formulæ $F_C(\gamma)$, $F_D(\gamma)$ are recursively given; similarly, for some DL role expressions R, Q the corresponding FOL open formulæ $F_R(\alpha, \beta)$, $F_Q(\alpha, \beta)$ are given.

An important feature of DL is that they are equipped with a formal calculus that allows to perform some inferential tasks [12]. The most important are (i) *consistency checking*: a concept description C is said to be *consistent* iff the corresponding FOL formula, $F_C(\gamma)$, is satisfiable; (ii) *subsumption*: a concept description C is said to *subsume* a concept description D (written $D \sqsubset C$) iff $\forall x.F_D(x) \Rightarrow F_C(x)$ is a FOL tautology.

In DL it is possible to define *knowledge-bases* (KB) as sets of subsumption assertions. It is said that a subsumption is entailed by a KB ($KB \models D \sqsubset E$) iff $F_{KB} \models \forall x.F_C(x) \Rightarrow F_D(x)$ in terms of FOL.[4] The consistency of a concept description C w.r.t. a (consistent) KB can be expressed as: $KB \models C \not\equiv \bot$.[5]

3.2 Conceptual Modeling in DL

To see how a database conceptual model can be described in terms of Description Logics, let's consider the fragment shown in fig.2 of a sample conceptual model.[6] The corresponding Description Logic KB is sketched in fig.3.

[4] Where F_{KB} is the set of FOL axioms containing $\forall x.F_{C_1}(x) \Rightarrow F_{C_2}(x)$ for each $C_1 \sqsubset C_2$ in KB.

[5] $C \equiv D$ stands for $C \sqsubset D \wedge D \sqsubset C$, and the special concept $\bot$ correspond to the *false* symbol in FOL. Similarly, the special concept $\top$ corresponds to the *truth* in FOL.

[6] Although, due to space limits, the formal semantics of the language used cannot be presented here, the meaning of the example should be quite intuitive and clear.

$Stud \sqcap Lect \equiv \bot$
$Stud \sqcap Prof \equiv \bot$
$Prof \sqcap Lect \equiv \bot$

$\exists hold.\top \sqsubseteq Prof$
$\exists hold^{-1}.\top \sqsubseteq Lect$
$Prof \sqsubseteq \forall hold.Lect$
$Eng\text{-}Prof \sqsubseteq Prof \sqcap \forall hold.Eng\text{-}Lect$
$CS\text{-}Prof \sqsubseteq Prof \sqcap \forall hold.CS\text{-}Lect$

$\exists attend.\top \sqsubseteq Stud$
$\exists attend^{-1}.\top \sqsubseteq Course$
$\exists participate.\top \sqsubseteq Stud$
$\exists participate^{-1}.\top \sqsubseteq Seminar$
$Lect \sqsubseteq \forall hold^{-1}.Prof \sqcap \exists^{=1} hold^{-1}$
$Seminar \sqcap Course \equiv \bot$
$CS\text{-}Lect \sqcap Eng\text{-}Lect \equiv \bot$
$Lect \sqsubseteq CS\text{-}Lect \sqcup Eng\text{-}Lect$

$Seminar \sqsubseteq Lect \sqcap \forall Participate^{-1}.Stud$
$Course \sqsubseteq Lect \sqcap \forall Attend^{-1}.Stud$

$CS\text{-}Lect \sqsubseteq Lect \sqcap \forall hold^{-1}.CS\text{-}Prof$
$Eng\text{-}Lect \sqsubseteq Lect \sqcap \forall hold^{-1}.Eng\text{-}Prof$
$CS\text{-}Course \equiv CS\text{-}Lect \sqcap Course \sqcap \forall attend^{-1}.CS\text{-}Stud$
$Eng\text{-}Course \equiv Eng\text{-}Lect \sqcap Course \sqcap \forall attend^{-1}.Eng\text{-}Stud$
$CS\text{-}Seminar \equiv CS\text{-}Lect \sqcap Seminar$
$Eng\text{-}Seminar \equiv Eng\text{-}Lect \sqcap Seminar$

$Stud \sqsubseteq \forall paricipate.Seminar \sqcap \forall attend.Course$
$CS\text{-}Stud \sqsubseteq Stud \sqcap \forall attend.CS\text{-}Course \sqcap \exists^{\geq 1} attend$
$Eng\text{-}Stud \sqsubseteq Stud \sqcap \forall attend.Eng\text{-}Course \sqcap \exists^{\geq 1} attend$

Fig. 3. Translation of the conceptual model into a KB.

In our paradigm, the use of KB like that presented in fig.3 must allow to infer, e.g., that a `CS-stud` (CS student) must `attend` only `CS-Course` (CS courses). Thus, according to CRCP, queries where a CS student is required to attend an engineering course (`Eng-Course`) must be immediately recognized as inconsistent, or, better, must be not even allowed to be built.

3.3 DL Reasoning and CRCP

Let's consider, for example, the query:

$$Q(x) \leftarrow \texttt{Stud}(x) \wedge \neg\texttt{Eng-Stud}(x) \wedge \texttt{attend}(x, y) \wedge \texttt{Course}(y) \qquad (1)$$

and consider the process needed to build it, according to to our paradigm. A step of the process could be conjoining $\neg\texttt{Eng-Stud}(x)$ to

$$Q(x) \leftarrow \texttt{Stud}(x) \wedge \texttt{attend}(x, y) \wedge \texttt{Course}(y).$$

Once query (1) is built, selecting `Course` for refinement must cause that it is evidenced that replacing $\texttt{Course}(y)$ with $\texttt{CS-Course}(y)$ would lead to an equivalent query: this is important, because the equivalence between the two forms conveys to the user more knowledge about the conceptual model.

On the other hand, we could try to build the same query by another approach, i.e., by adding the association `attend` to $Q(x) \leftarrow \texttt{Stud}(x) \wedge \neg\texttt{Eng-Stud}(x)$. In this case, according to CRCP, $\texttt{attend}(x, y) \wedge \texttt{CS-Course}(y)$ must be directly proposed to the user, and not the less informative $\texttt{attend}(x, y) \wedge \texttt{Course}(y)$. Of course this is not an arbitrary action, because, as before:

$$Q(x) \leftarrow \texttt{Stud}(x) \wedge \neg\texttt{Eng-Stud}(x) \wedge \texttt{attend}(x, y) \wedge \texttt{CS-Course}(y)$$

and

$$Q(x) \leftarrow \texttt{Stud}(x) \wedge \neg\texttt{Eng-Stud}(x) \wedge \texttt{attend}(x, y) \wedge \texttt{Course}(y)$$

are equivalent.

All these reasoning services can be performed by means of the basic inferential services of *consistency checking* and *subsumption* provided by a DL. In order to do this, it is necessary to represent the query in DL. For example, the query (1) is represented by:

$$Stud \sqcap \neg Eng\text{-}stud \sqcap \exists attend.Course \tag{2}$$

It is quite easy to verify that:

$$KB \models$$

$$Stud \sqcap \neg Eng\text{-}stud \sqcap \exists attend.Course \equiv Stud \sqcap \neg Eng\text{-}stud \sqcap \exists attend.CS\text{-}course$$

where KB is the knowledge base of fig.3.

Of course, this is just one example of reasoning; in particular it is relevant for the functionality of *relation selection* seen in subsection 2.2. More systematically, the kinds of checks corresponding to each of the functionalities introduced in subsection 2.2 are sketched below:

- *Query replacement.* It is enough to maintain a representation of the query in term of DL, like expression (2), and to *classify* it in the *taxonomy* implied by the KB. That is, the most specific subsumers, $MSS(Q)$, and the most generic subsumees, $MGS(Q)$, of Q must be found in KB, in order to update the generalization and the specialization lists, respectively.
- *Relation selection.* Here the task is more complex, and require two steps:
  1. list all the *compatible relationships* by testing, for each relation name R in the KB, if $Q \sqcap \exists R.\top$ is consistent (and similarly, for the inverse relation test if $Q \sqcap \exists R^{-1}.\top \not\equiv \bot$);
  2. for each compatible relationships R (or R^{-1}) determined in step 1, define the relevant range as $MSS(\exists R^{-1}.Q)$ (or $MSS(\exists R.Q)$).
- *Propositional combination.* Let's consider the case of `AND` combinations. Assume that q is the sub-query selected in Q. Let $Q_{q/q'}$ denote the new query obtained from Q by replacing the sub-query q with q' in Q. What must be checked is CRCP of $Q_{q/(q \sqcap C_i)}$ w.r.t. Q, for all the $C_i \in KB$. It is worth noticing that only the specialization list is meaningful during this operation. Several strategies can be adopted in order to reduce the search of the C_i, as, e.g., following a top-down search in the taxonomy, with early prunings when inconsistent cases are found, and no further search when a relevant C_i is reached (in this case $Q_{q/(q \sqcap C_i)}$ will be the closest to Q, along the considered path, due the top-down search). The top-down strategy gives good results because, in practical cases, the conceptual model contains disjoint classes at a quite high level in the taxonomy, allowing for a good amount of pruning. For `OR` combinations the general approach is dual, even if the dual (bottom-up) strategy is not as efficient as the top-down strategy for the conjunctions.
- *Negation.* The case of negation is quite easy: it is enough to check if the query ($Q_{q/\neg q}$) will still be consistent.
- *Refinement.* In this case the situation is similar to the case of propositional combination, seen above, but a bidirectional search must be performed. The CRCP must be checked on all the possible Q_{q/C_i}. A possible heuristics is to

start from $MSS(q)$ and $MGS(q)$, and search upward and downward respectively, in order to find the generalization and the specialization lists. Also in this case, reaching a relevant C_i halts the search.

Of course, the paradigm here proposed can be implemented only in presence of a DL automatic reasoner, capable to manage KB with circular definitions (or general axioms). By the example it is clear that the DL language required must allow to express conjunctions, disjunctions, negations, qualified quantifications and number restrictions, and inverse roles; that is $\mathcal{ALCNIF}$. iFaCT [13], already capable to perform reasoning over the $\mathcal{ALC}_{\mathcal{R}^+}\mathcal{HIF}$, has recently been improved with the capability of reasoning over qualified number restrictions. It is, therefore, suited to be used to implement the paradigm here presented.

4 A Prototypical System

Starting from the ideas presented in this paper and from the availability of iFaCT, we have implemented a first prototype for testing our paradigm. The system uses iFaCT, at which the functionalities described in subsection 2.2 and subsection 3.3 have been added by us. A visual interface (see fig.1) has been implemented, in order to test on users the impact of the semantic based interaction. More details on the prototype are given in [6].

The system architecture is based on a client-server paradigm. The visual interface is written in Java in order to allow the highest portability. The reasoning services run on Allegro Common Lisp, possibly on a remote machine. Socket channels provide the low level communication between the interface and the reasoning server. In [6] some more particulars are presented.

5 Conclusions

The most important aspect of the paradigm for querying databases presented in the present paper consists on the fact that the use of knowledge based reasoning is proposed, in order to help the user in the task of building sophisticated queries through an iterative process of successive refinements and generalizations, that allow to progressively discover interesting meanings and consequences hidden in complex conceptual model descriptions, and prevents, at the same time, from submitting inconsistent queries.

The paradigm, and its first prototypical implementation, can be considered a first step toward a practical and experimental activity oriented to face several issues about the application of knowledge based systems in the field of database access and interfaces, as mentioned in section 1 and, in a more complete way, in [4]. How to translate the conceptual representation of the queries, sketched in this paper, into SQL queries, in order to access real DBMS, is detailed in [3].

Some more efforts are now necessary to complete the development of the prototype. At the same time, we plan to analyze and code some complex examples of conceptual models, in order to perform some tests with unskilled users, and

try some comparisons with other VQS, like, for example QBD, [10] and VISIONARY [1]. In any case, we foresee that the comparison will not be straightforward, because our approach stress semantic aspects, while others systems emphasize the visual interaction.

References

1. F. Benzi, D. Maio, and S. Rizzi. VISIONARY: a viewpoint-based visual language for querying relational databases. *Journal of Visual Languages and Computing*, 10:117–145, 1999.
2. Alex Borgida. Description logics for data management. *IEEE Transactions on Knowledge and Data Engineering*, 5(7), October 1995.
3. Paolo Bresciani. The challenge of integrating knowledge representation and databases. *Informatica*, 20(4):443–453, December 1996.
4. Paolo Bresciani. Some research trends in knowledge representation and databases. In F. Baader, M. Buchheit, M. A. Jeusfeld, and W . Nutt, editors, *Working Notes of the ECAI-96 Workshop "Knowledg e Representation Meets Databas (KRDB-96)"*, pages 10–13, Budapest, August 1996.
5. Paolo Bresciani and Enrico Franconi. Description logics for information access. In *Proc. of "Giornata di Lavoro su accesso, estrazione ed integrazione di conoscenza - V Convegno AI*IA"*, September 1996.
6. Paolo Bresciani, Michele Nori, and Nicola Pedot. QueloDB: a knowledge based visual query system. In *Proceding of IC-AI: International Conference of Artificial Intelligence*, Las Vegas, June 2000. Forthcomming.
7. M. Buchheit, M. A. Jeusfeld, W. Nutt, and M. Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.
8. D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of the 4^{th} International Conference on Deductive and Object-Oriented Databases, DOOD'95*, Singapore, December 1995.
9. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.
10. T. Catarci, S.K. Chang, M.F. Costabile, S. Levialdi, and G. Santucci. A graph-based framework for multiparadigmatic visual access to databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):455–475, 1996.
11. T. Catarci, M. F. Costabile, S. Levialdi, and C.Batini. Visual query systems for databases: a survey. *Journal of Visual Languages and Computing*, 8(2):215–260, April 1997.
12. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2^{nd} International Conference on Principles of Knowledge Representation and Reasoning*, pages 151–162, Cambridge, MA, 1991.
13. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
14. M. M. Zloof. Query–by–example: A database language. *IBM System Journal*, 16(4):324–343, 1977.

PERFORMANCE ANALYSIS OF WEB DATABASE SYSTEMS

Yuanling Zhu, Kevin Lü

SCISM, South Bank University
103 Borough Road
London SE1 OAA
Email: zhuy@sbu.ac.uk

Abstract. Currently, Web database systems are widely used to construct Web sites for their excellent capability of providing on-line information. In this paper, we analyse the workflow of a Web database system dealing with a Web page request. After forwarding the data request to database servers, Web servers must wait until all or part of query result is sent back by the database server before proceeding to next step. This is a typical serial queuing system, which is not easy to model except the service-time distribution is exponential. We have classified the different cases and given an approximate method to model a Web database system. As the service time of database servers is a primary factor in determining the input characteristics of Web servers, it is very important to investigate the relationship between database servers and Web servers.

1 Introduction

The World Wide Web (WWW) is an excellent media for the dissemination of information. The ability to populate Web sites with content derived from large databases has become the key to building enterprise Web sites [Flo98]. Web databases, which provide Internet-based accesses to remote users, enable companies to extend their services to a wider range of users through the Internet. As business transactions are increasingly executed by means of the Web, many efforts have been made on improving the connectivity between databases and the Web [Vet99]. Traditional RDBMS vendors manage to make their products act as Web databases which can be accessed from the Web [Ora99] [MS99].

Unfortunately, most of current Web database products have been criticised for poor performance, which will bring about serious consequences, such as losing clients, revenue reduction and damaging a company's reputation. To solve the problem, a number of projects that aim to improve the performance of Web databases have been carried out. Recently, performance issues of Web servers become the top concerns in the research community [Bar98][Spe99].

In this paper, a Web database system refers to an integrated system of Web servers and database servers, which enables users to access on-line information in a platform-independent manner through Web browsers. In such a system, database servers play an important role in determining the overall system performance, which is neglected in most studies on Web performance. As Web servers depend on the availability of

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 805-814, 2000.

data provided by database servers, different types of database applications will result in various workloads on Web servers. Consequently, the performance metrics, such as throughput and response time, should be evaluated by partitioning the workload into several classes according to the application type. This paper presents an analytical performance model of Web servers with the consideration of database involved applications. Based on the model, the issues related to performance metrics of Web database systems are also addressed.

The remainder of the paper is organised as follows: Section 2, introduces the related work. Section 3, proposes the performance model of Web database systems. Section 4, analyses the performance of a Web database system by an example and suggests performance optimisation strategies. Section 5, concludes the paper and discuss pending research issues.

2 Related Work

During the last few years, several studies have been carried out to model and analyse Web performance problems based on queuing network theories. [SLO96] modelled the Web server and the Internet as an open single class queuing network, from which they derived several performance results for Web servers on the Internet. This work focused on the case of requesting only static Web pages, of which the contents remains the same unless some changes are made. Moreover, it assumed the sizes of requested files are distributed exponentially, which means it did not classify the workload on the Web server according to its characteristic. [MA98] modelled the Web servers as an open multiple-class queuing network model. Similar to the studies in [SLO96], it did not consider the case of dynamic Web page requests, of which the contents is extracted from a backend database server or created by an application. As a result, different classes in the model only corresponded to HTTP requests of different sizes. In both models, the impact of database servers is not considered in system performance modelling.

However, as Web applications requiring database accesses become very common in today's Web-based computing, it is thus very important to understand the performance problems of Web systems with consideration of database servers.

3 Performance Model

3.1 Workload Classification

The first step in any performance evaluation study is to understand and characterise the workload [MA98]. The workload can be categorised into classes based on resources usage. Each class comprises requests that are similar to each other concerning resource usage. A Web server may sometimes receive requests for static Web pages with sizes below 1KB. It is also possibel that a user may submit a request embedding a complex query and the amount of requested data could be up to 100MB.

The system resource usage is quite different for processing these two types of applications. While the former one may require a negligible amount of time at the Web server, the complex query may take several seconds of the database server to compute the result. Because of the heterogeneity of Web applications, it is not sufficient to represent the workload of Web database servers by a single class.

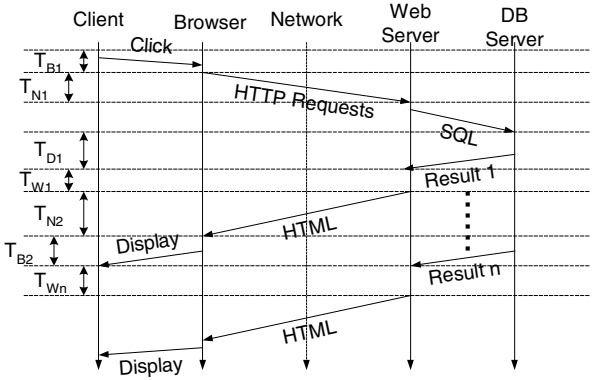


Fig. 1. Time delayed by each component of a web databases system

Before characterising the workload of a Web database system, components involved in a Web database application processing must be checked first: when a Web server receives a request with a query to databases, it parses the request and forwards the embedded query to databases. The database server executes the query and sends the result back to the Web server. The Web server then replies the client with a response consisting of information just retrieved from the database in HTML format. The response time of a request is determined by the request's residence time at all the four components. Figure 1 depicts the time delay caused by each component, including the browser, the network, the Web server, and the database server.

It can be inferred that the workload on a Web database system can be characterised by the system resources consumption on Network, Web servers and database servers. According to our tests, the size of the requested document is the main factor to networking delays [Zhu00]. Similarly, large Web pages will demand much more system resources of Web servers. Meanwhile, database applications are often characterised by the complexity degree of the query.

Class	**Description**	**Load on DB Servers**		**Load on Web Servers**		**Load on Network**
		CPU	**Disk**	**CPU**	**Disk**	
1	Complex query with small result size	High	Low	Low	N/A	Low
2	Complex query with large result size	High	High	High	N/A	High
3	Simple query with small result size	Low	Low	Low	N/A	Low
4	Simple query with large result size	Low	High	High	N/A	High
5	No query, small file size	N/A	N/A	Low	Low	Low
6	No query, large file size	N/A	N/A	High	High	High

Table 1. Workload classification in a Web databse system

Table 1 depicts a typical classification of the workload on a Web database system. The workload of a Web database system is categorised into six classes of similar requests based on the resource usage on the primary components. For class 1, a request of this type embeds complex queries with small result sizes. It demands high CPU but low I/O time consumption on the database server. Consequently, it requires low CPU time of the Web Server to process small amount of data and produces low load on the network. Similarly, a request of class 6 demands no resource on database servers but requires high CPU and I/O time consumption and produces high load on the network.

3.2 Performance Model

Figure 2 illustrates the model that is used to describe a Web database system. In fact, it is an open multiple-class queuing network model. There are four load-independent queues: the first one is the queue of incoming requests on Web servers. The second queue is for the file requests and the third queue is for the database-involved requests. The last one depicts the response queue waiting for being sent back through the network.

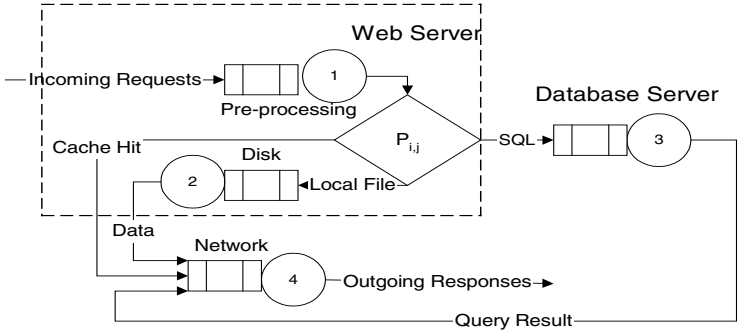


Fig. 2. Queuing Network model for Web databases systems

Web page requests arrive at the Web server with the average arrival rate $\vec{\lambda}(\lambda_1, \lambda_2, \cdots, \lambda_r, \cdots, \lambda_R)$, Where λ_r represents the arrival rate of the workload of class *r*. Upon receiving the request, the Web server will check its cache first. If the requested data is available there, then the cached data is passed to the network queue waiting for being sent back to the client. Otherwise, the request will proceed to the disk queue of the Web server if it only requests static Web pages, or proceed to the database queue if it requires access to databases. After the requested data is fetched from local disk or retrieved from databases, it goes to the network queue.

When a request demands access to databases, Web servers must wait until all or part of the query result is sent back by the database server before proceeding to the next step. In order to present the impact of database servers on the overall performance of a Web database system, the Web database system is modelled as an open queuing network with First Come First Service (FCFS) discipline, multiple classes, and general service pattern at queuing centres containing a single server. This

is a typical serial queuing system, which has exact solutions only when the service time distribution is exponential. The maximum entropy method is adopted to solve this kind of queuing network model for its numerical accuracy and low computational cost [Kou85] [Kou93].

Different classes in the model are categorised according to the characteristics of applications as mentioned in the last section. Six classes of the workload and four queues are considered in our model. Assume the following notation for class *r* requests at queue *i*:

$\lambda_{i,r}$	Mean arrival rate
$C^2_{ai,r}$	Squared coefficient of variation of inter-arrival times
$\mu_{i,r}$	Mean service rate
$C^2_{si,r}$	Squared coefficient of variation of service times
$\rho_{i,r} = \frac{\lambda_{i,r}}{\mu_{i,r}}$	Load on server
$p_{j,t;i,r}$	Probability that a request of class *t*, having completed service at queue *j*, then changes to of class *r*, and queues for service at queue *i*

Let $\lambda_r (r = 1,2,..,6)$ be the arrival rate of class *r* requests. The average arrival rate is computed as:

$$\lambda_r = \lambda \times PercentOfClass\_r \tag{1}$$

Let $\lambda_i = \sum_{r=1}^{R} \lambda_{i,r}$ be the total mean arrival rate at queue *i*, the probability that an arbitrary request at queue *i* is of class *r* can be computed as:

$$\pi_{i,r} = \frac{\lambda_{ir}}{\lambda_i} \tag{2}$$

p_{ji} is the probability of a request going from queue *j* to queue *i*:

$$p_{ji} = \sum_{r=1}^{R} \sum_{t=1}^{R} \pi_{j,t} p_{j,t;i,r} \tag{3}$$

The coefficients of variation of the departure process at queue *i* is:

$$C^2_{di} = \frac{(1-\rho_i)\lambda_i}{\sum_{j=0}^{M} \lambda_j p_{ji} \Big/ [2 + p_{ji}(C^2_{dj} - 1)]} + \lambda_i \sum_{r=1}^{R} \frac{\rho_{i,r}(C^2_{sir} + 1)}{\mu_{i,r}} + 2\rho_i(1-\rho_i) - 1 \tag{4}$$

Where $\rho_i = \frac{\lambda_i}{\mu_i}$ denotes to the load on queue *i*.

Then the coefficients of variation of the inter-arrival processes, i.e., the differential of requests transferring from queue *i* to queue *j* is:

$$C_{dj,i}^2 = 1 + \sum_{r=1}^{R} \sum_{t=1}^{R} \pi_{j,t} p_{j,t;i,r} (C_{dj}^2 - 1) \tag{5}$$

The squared coefficient of variation of arrivals at queue *i* is:

$$C_{ai}^2 = \left(\lambda_i \Big/ \sum_{j=0}^{M} \frac{\lambda_j p_{ji}}{C_{dj,i}^2 + 1} \right) - 1 \tag{6}$$

The squared coefficient of variation of service time of queue *i* is:

$$C_{si}^2 = \frac{\mu_i}{\rho_i} \sum_{r=1}^{R} \frac{\rho_{i,r} (C_{si,r}^2 + 1)}{\mu_{i,r}} - 1 \tag{7}$$

Finally, the mean number of requests in the queue *i* is given by:

$$\overline{n_i} = \frac{\rho_i}{2} \left(1 + \frac{C_{ai}^2 + \rho_i C_{si}^2}{1 - \rho_i} \right) \tag{8}$$

Let $\overline{n_{i,r}}$ be the mean number of requests of class *r* in queue *i*, it can be calculated by:

$$\overline{n_{i,r}} = \pi_{i,r} (n_i - \rho_i) + \rho_{i,r} \tag{9}$$

Since a request's waiting time at the queue does not depend on its class, the mean waiting time per class *r*, is (by Little's Law):

$$W_r = \overline{n} \Big/ \lambda - 1 \Big/ \mu \tag{10}$$

And the mean response time per class *r* is:

$$R_r = \overline{n} \Big/ \lambda + 1 \Big/ \mu_r - 1 \Big/ \mu \tag{11}$$

4. Performance Analysis

An example is presented in this section to illustrate how to use the model to analyse the performance of a Web database system. For simplicity, cache is also not considered in this example. The pre-processing time of each request is nearly independent to workload classes and is negligible when compared to the time spent on

disks, databases or network [Zhu00]. A simplified version of the model is shown in Figure 3.

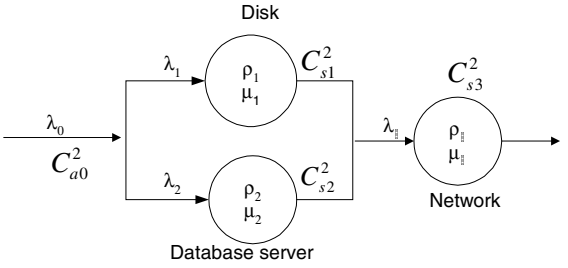


Fig. 3. Three-stage open queuing network model

Table 2 shows the initial parameters of the model according to the class of workload. The service time deferential refers to the squared coefficient of variation of service times of each component. It is reasonable that the service time deferential would be higher if the service time is longer.

Class	Mean service rate			Service time deferential			Load1	Load 2
	Disk	DB server	Network	Disk	DB server	Network		
1		50	500		3	1	5%	5%
2		20	50		4	3	5%	0%
3		200	500		1	1	40%	35%
4		40	50		2	3	10%	10%
5	200		500	1		1	25%	40%
6	40		50	2		3	15%	10%

Table 2. Resource usage of each class of workload on queue *i*

With these input parameters and the model proposed in the last section, the performance metrics of a Web database system, such as response time, throughput, bottleneck, and maximum capacity, can be predicted. Figure 4 and figure 5 depict the queue length of each resource at the different request arrival rates with the workload shown in table 2.

It can be observed that the queue length does not increase linearly with the arrival rate. When the system nearly reaches its maximum capacity, the queue length increases sharply and so does the response time of the system become unacceptable. The bottleneck of the system can also be detected from the figures. It can be observed that the database server is the bottleneck in figure 4 and the network is the bottleneck in figure 5. It is understandable that the maximum capacity of the system is determined by the maximum capacity of the bottleneck resource. As shown in Figure 4, although the network can serve much more requests, the database server has already reached its maximum capacity. So in this case, the maximum capacity of the database server represents the maximum capacity of the Web database system. In the case of Figure 5, the network become saturate before the database server reaches its maximum capacity. So the network is the main factor to determine the maximum system capacity this time.

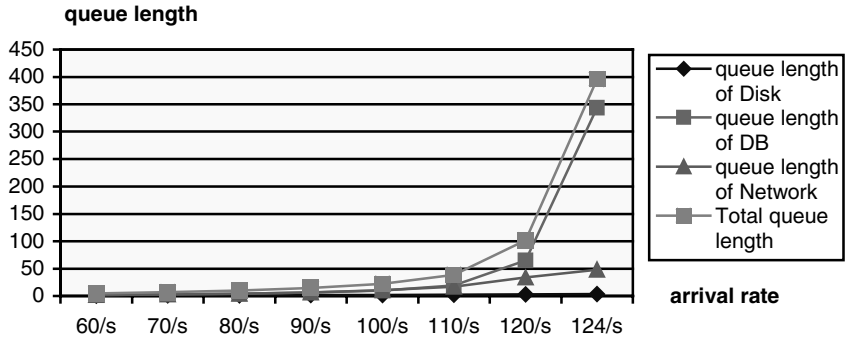


Fig. 4. Queue length at different arrival rate with workload 1

Figure 4 and Figure 5 also demonstrate that different workload distributions affect the system performance greatly. The Web database system under the workload of Figure 5 can serve much more requests than the system under the workload of Figure 4. So it is indispensable to analyse the workload of a Web database system when predict its performance and capacity.

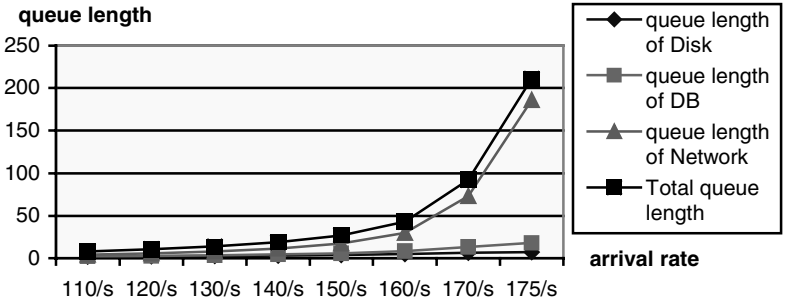


Fig. 5. Queue length at different arrival rate with workload 2

Furthermore, in order to illustrate the importance of the database server in a Web database system, database-related workload should be investigated. It is possible that a database server can complete a request of class 1 in the same amount of time as to complete a request of class 4. Although these two requests impose the same load on the database server, they are treated differently in a Web database system. When the database server serves a request of class 4, it can send some data to the Web server before having done the entire request. But in the case of a request of class 1, the data will not be available until the database server computes a lot and finally obtains the required data. In the former situation, Web servers can work in parallel with database servers. There is an overlap between the time spent on the Web server and the database server, especially in the case of large amount of data being extracted from databases.

Class	Load 1	Load 2	Load 3	Load 4	load 5	load 6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0.26	0.31	0.36	0.41	0.45	0.5
4	0.34	0.33	0.32	0.31	0.3	0.29
5	0.3	0.255	0.21	0.165	0.13	0.085
6	0.1	0.105	0.11	0.115	0.12	0.125

Table 3. Different Workload with the same load imposed on the database server

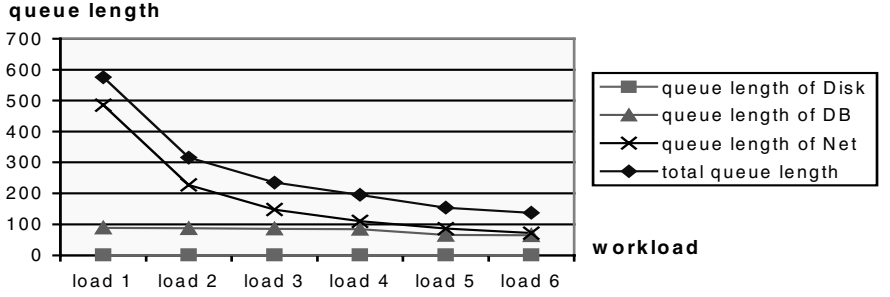


Fig. 6. Queue length with different workload

As mentioned before, the model in this paper is a serial queuing network, which means a request must complete the entire request on the first stage before going to the second stage. It implies that all the data must be read from disks or retrieved from databases before going to the network queue. When the size of requested data is very big, the Web server will wait for a long time before it can process the request. In fact, some parallelism exists between the database server and the Web server. In order to simulate this feature by the model, a request of class 4 can be seen as several requests of class 3. Table 3 lists some workload distribution with the same load on the database server and Figure 6 shows the queue length of each resource under these workloads. When some requests of class 4 was replaced by some requests of class 3 with the same load on the database server, the queue length of the database server decreased. Moreover, the queue length of the network decreased too. It is attribute to the less workload on the network and higher resource utilisation.

5 Conclusion

Various issues related to Web database systems have been investigated in this study. This type of system has been widely used to provide on-line information service in the Internet. However, the performance issues of Web database systems have received little attention in the database research community. In this paper, components involved in a web database transaction processing are examined and their relationships to the performance of Web database systems are described. With the consideration of the aspects of Web database applications, workload on a Web

database system is categorised into multiple classes with different resource usage. Based on this, a Web database system is modelled as a multiple-class open queuing network. With this model, performance metrics of a web database system can be approximately calculated.

Based on our analyses, the performance of a Web database system might be improved if the parallelism between the Web server and the database servers are exploited. This paper studies the workload of classes with simple queries and suggests that the Web server can send back part of data while the database server is still working on the remaining. It is different in the case of workload of classes with complex query, because the data will not be available at the beginning. To solve the problem, some work focused on the optimisations that exploit the structure of the Web site [Flo99]. A complex query can be decomposed into smaller chunks based on a declarative site specification and benefit from utilising cache of recent query result. Our future work will include the study of these classes of workload.

6. References

[Bar98] Barford, P., *Generating representative Web workloads for network and server performance evaluation.* Proceedings of the ACM SIGMETRICS'98 Conference. ACM, June 1998.

[Flo98] Florescu, D., 1998. *Database Techniques for the Wide Web: A Survey.* SIGMOD Record 27(3): 59-74(1998)

[Flo99] Florescu, D., 1999. *Optimisation of Run-time Management of Data Intensive Web Sites.* Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999

[Kou85] Kouvatsos, D.D., 1985. *Maximum Entropy Methods for General Queueing Networks.* In D. Potire, editor, Modelling Techniques and Tools for Performance Analysis, Pages 589-608. North-Holland, Amsterdam, The Netherlands, 1985.

[Kou93] Kouvatsos, D.D., Denazis S.G., 1993, *Entropy Maximised Queueing Networks with Blocking and Multiple Job Classes*, Performance Evaluation 17(3): 189-205 (1993)

[MA98] Menasc, D., Almeida, V., 1998. *Capacity Planning for Web Performance,* Prentice Hall Publisher

[Ora99] *Oracle Application Server Technical Document*, 1999. http://technet.oracle.com

[MS99] *Application Server Page Technical Document*, http://msdn.microsoft.com/, 1999

[Slo96] Slothouber, L., 1996. *A model of Web Server Performance.* http://louvx.biap.com/WebPerformance/modelpaper.html

[Spe99] *SPECweb99 Release 1.01 Specification,* http://www.sprc.org/osg/web99/, 1999

[TPC99] *TPC BENCHMARK™ W (Web Commerce) Draft Specification*, http://www.tpc.org/, November 19, 1999

[Vet99] Vetter R., 1999. *Web-based enterprise computing,* Computer 5/1999.

[Zhu00] Zhu Y., Lu, K.J. 2000, *Web Databases and Related Performance Issues*, Submit for publication.

WDBQS: A Unified Access to Distant Databases Via a Simple Web-Tool

Christian Sallaberry [1], Eric Andonoff [2], and Nicolas Belloir [2]

[1] Université de Pau et des Pays de l'Adour, UFR Droit & IAE, 64000 Pau, France,
christian.sallaberry@univ-pau.fr
[2] IRIT/UT1, Université Toulouse 1, 1 place Anatole France, 31042 Toulouse Cedex, France
ando@irit.fr

Abstract. Within organisations, Information Systems are characterised by heterogeneity. The main management tools are different Relational and Object Oriented DataBase Management Systems. Therefore, variety, complexity and lack of integration make it difficult for casual users to query them. This paper presents WDBQS (Web DataBase Query System); a Web-interface for querying distant Relational and Object Oriented DataBases. This system, dedicated to casual users, proposes simple Web mechanisms to navigate through database schemas, build queries and display results.

1 Introduction

We have collaborated with Local Communities (LC) to produce WDBQS (Web DataBase Query System), an ongoing project that aims to develop a tool intended for casual users for distant relational and object-oriented database querying (RDB and OODB). LC handle a set of Information Systems managed by Relational DBMS and Object-Oriented DBMS (RDBMS and OODBMS). These IS have been developed and managed independently; they are based on different software technologies and run on various hardware platforms. WDBQS offers to the LC users a common language to query the databases implementing the different IS. There is no need to develop approaches like DataWareHouse or MultiDataBase. Indeed, LC users just need a tool to query, one after another, the different distant and independent RDB and OODB.

So, WDBQS is a relational and object-oriented database query system. Its main characteristics are:

- ODMG data model,
- common access to distant RDB and OODB,
- accessible via the Web in an intranet architecture and advocates a navigational query formulation mechanism,
- reuse of data and systems in order to take advantage of existing resources and be able to rely on proven technologies (RDBMS and OODBMS),

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 815–824, 2000.

- extendibility in order to access any new database, either achieved through the corresponding DBMS drivers already integrated in WDBQS or through the integration of a new DBMS drivers in WDBQS,
- specific interfaces so as to manage distant databases accessed by WDBQS, define queries and display results.

WDBQS makes accessible heterogeneous DBMS via a unique platform. It is based on the ODMG model and proposes a Web-like query language, called WDBQL (Web DataBase Query Language). WDBQL is intended for casual users i.e. users who do not have any knowledge in computer-science and consequently in databases and database query languages such as SQL and OQL. WDBQL proposes a Web-interface and implements a Web-like i.e. navigational mechanism to formulate queries. Such a mechanism is easy to use for casual users and really helps them when query formulation.

The remainder of the paper is organised as follows. Section 2 presents the WDBQS architecture. Section 3 introduces the WDBQS models and a running example. Section 4 describes the WDBQL querying principles and presents the WDBQL interface. It also gives a comparison with related works. Section 5 concludes the paper.

2 WDBQS architecture

WDBQS proposes a client-server architecture described in figure 1. It is composed of WDBQL, WDBQS Server (WDBQSS), Host DataBase Server (HDBS) and Remote DataBase Server (RDBS).

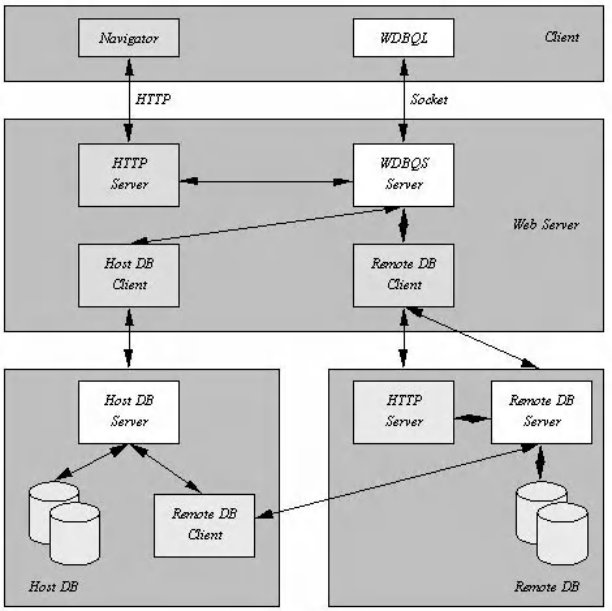


Fig. 1. WDBQS architecture

HDBS stores in a special database called the Host DataBase, a meta-schema which describes the schemas of all the distant RDB and OODB. The model of this Host DataBase is the ODMG one [9]. The distant RDB and OODB schemas are stored as instances of the meta-schema. HDBS uses a schema extractor and an ODMG wrapper to first extract the schema of a distant database and to then convert it as instances of the meta-schema. There are as many schema extractors as distant databases and as

many wrappers as distant database distinct models. The interest of such an approach is that the number of accessible distant databases is easily extendable.

RDBS is the server of the distant database. It executes queries and returns the corresponding result. Two situations are possible: RDBS integrates a Web module (e.g. O2Web, OracleWeb) or not. If the Web module exists, RDBS displays the results by itself in a new client Web navigator window. If not, RDBS returns a set of data which are converted by the WDBQL Data Translator into the WDBQL visual format. RDBS also notifies HDBS of any database schema updates. HDBS then starts again schema extraction and conversion.

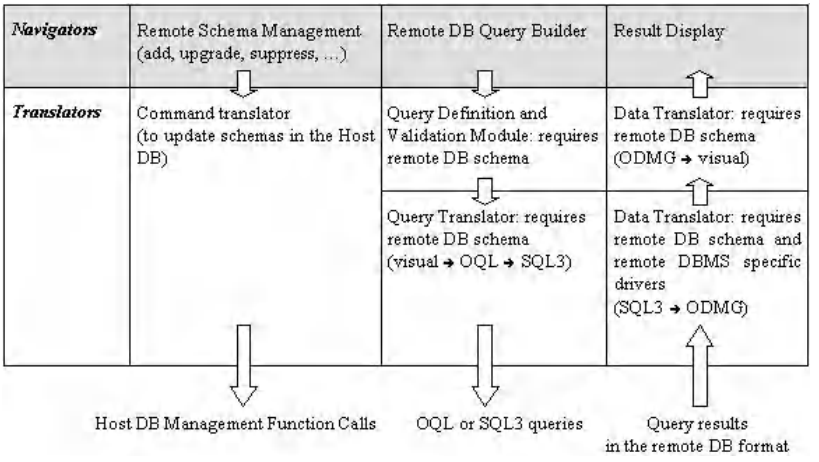


Fig. 2. WDBQS: navigators and translators

WDBQL offers two main navigators: one for remote database schema management and another for remote database querying. Each navigator translates users visual queries respectively into database management function calls or OQL (or SQL3) queries for WDBQSS. A third navigator possibly displays results via an HTML wrapper. The WDBQL querying navigator uses the Query Definition and Validation module to display the schema of the queried remote database and to help the user when query formulation. The WDBQL querying navigator also uses the Query Translator module to translate WDBQL queries into OQL ones. The query Translator module principally uses the translation algorithms presented in [10].

WDBQSS manages communication through all the modules of the system. The working of the system is the following (cf. Fig. 3.).

One user connects with WDBQSS via the HTTP protocol. WDBQSS sends back an HTML page integrating WDBQL to the client (Java applets). WDBQL displays the list of all the remote databases which can be queried. The user selects a database. The name of the selected database is returned to WDBQSS via a socket. WDBQSS gets the corresponding database schema in the HDB and sends it to WDBQL in a text format file using a socket once again. WDBQL interprets this data file and displays the data base structure to the user. Then the user formulates a query on his client machine and submits it to WDBQSS. The query is then shipped to RDBS to be exe-

cuted. RDBS either displays the result of the query in a new navigator (if it has its own Web module) or returns a set of data which are converted by the WDBQL Data Translator into the WDBQL visual format.

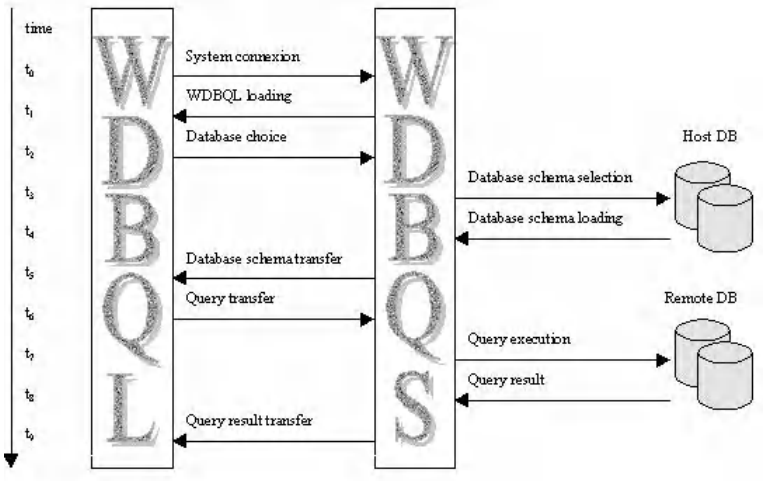


Fig. 3. Client (WDBQL) / Server (WDBQSS) communications

WDBQS architecture can be compared to that of GEM/WeBUSE [15]. The GEM/WeBUSE context is quite the same as the WDBQS one. The main difference is that GEM/WeBUSE proposes a remote GEM server for each distant database system. There is no central host database and HDBS but there are many remote meta-schema descriptors. This approach is not suitable in WDBQS as it makes too many communications (with the remote DBMS) when formulating queries. Moreover, in our context we are not allowed to use disk space within remote database servers.

3 The WDBQS models

We give here the *County Council* database example which manages boroughs, town halls, citizens, elected representatives and clerks. Because of space limitation, we only give an implementation of this example in a remote object-oriented database. O2 is chosen as object-oriented database system.

```
class Borough type tuple
( Bcode : integer,
  Bname : string,
  Bcity : string,
  Mayor : Elected,
  Deputies : set(Elected),
  Counselors : set(Elected),
  MainTownHall : TownHall,
  SecondaryTownHalls : set(TownHall) )
end;
```

```
class Person type tuple
( Pcode : integer,
  Surname : string,
  Firstnames : set(string),
  Address : tuple( street, zip, city : string ),
  Married : Person,
  Birthdate : Date )
end;
```

```
class Elected inherits Person type tuple
( Arrival : Date,
  Grade : string )
end;
```

WDBQS supports two levels of data representation: the database level and the user level. We now present the two level of data representation.

3.1 The database level

The database level of WDBQS stores the description of the different remote RDB and OODB schemas as instances of a meta-schema. This meta-schema is described using the standard ODMG-93 [9]. We choose the ODMG model because it generalises the RDB and OODB concepts. Besides, it is chosen as a pivot model in several federated DBMS.

The meta-schema is composed of a set of meta-types implementing the main ODMG model concepts. The specified meta-types are the Base, Type, Property, TypeProperty, BasicTypeProperty, MultiTypeProperty, StructuredTypeProperty, Operations and Relationship ones. The meta-schema and its corresponding types are detailed in [5].

3.2 The user level

The user level allows interpretation of the meta-schema within a more simple representation model. The latter is described using a graph model based on the node, link and anchor notions. Such a model obviously better suits casual users.

Each node has a name and is either terminal or non-terminal: terminal nodes are not connected to others by a link, while non terminal nodes are at least connected to one another by a link. The database model and the user model are perfectly matched as we can see in the following table:

Database Model	**User Model**
Base	Non-terminal node
Type	Non-terminal node
Property - Mono-valued - Multi-valued - Structured	 Terminal node Terminal node Non-terminal node
Operation	Terminal node
Relationship - Mono-valued - Multi-valued	 Non-terminal node Non-terminal node

Table. 1. Correspondence between the Database Model and the User Model

The instances of the meta-types *Base*, *Type* and *Relationship* correspond to non-terminal nodes in the user model while instances of *Operation* correspond to terminal

nodes. Instances of *BasicTypeProperty* (mono-valued property) and *MultiType-Property* (multi-valued property) are represented as terminal nodes while instances of *StructuredTypeProperty* (structured property) correspond to non-terminal nodes. Links are defined either between non-terminal nodes or between a non-terminal node and a terminal node. The figure below partially illustrates that correspondence with respect to the running example.

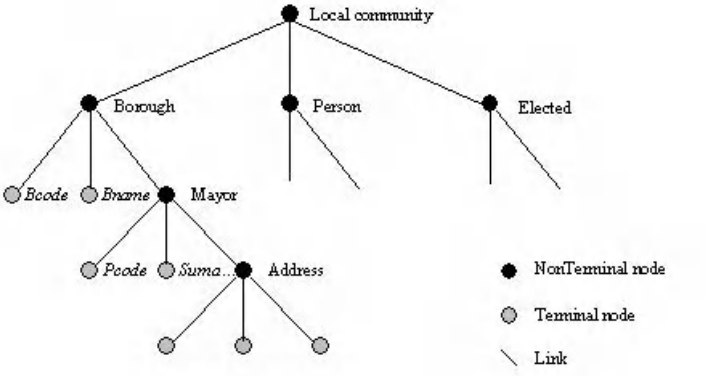


Fig. 4. Part of the County Council database described with the user model

4 WDBQL: a query language intended for casual users

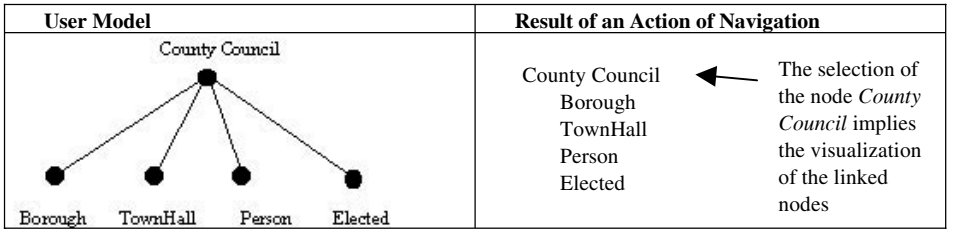
WDBQL is a Web-based query language intended for casual users. It allows the expression of queries from a Web interface using a navigational language.

4.1 The querying principles

WDBQL supports the user model. That means that the queried database schema is visualised as a set of nodes, links and anchors. A WDBQL query is a set of actions on nodes of the user model. We distinguish two types of actions which are navigation and selection.

Navigation only concerns non-terminal nodes. It is possible thanks to the anchors associated to each non-terminal nodes of the database. The name of a node is used as the corresponding anchor flag. The selection of an anchor visualises all the nodes linked to the node corresponding to the anchor. We illustrate below the navigation in the *County Council* database.

The *County Council* database is composed of the types *Borough*, *TownHall*, *Person* and *Elected*. When selecting the *County Council* database, these four types are visualised as shown below: Then the user can go on its navigation selecting for example the non-terminal node Borough. All the nodes linked to the node Borough are then visualised.



Selection concerns both terminal and non-terminal nodes. It has different semantics which depend on the type of the node. We distinguish four types of nodes: mono-valued terminal node, multi-valued terminal node, mono-valued non-terminal node and multi-valued non-terminal node.

The following tables present the operators and operands which can be used to define selection predicates. Because of space limitation, we only give below the table for multi-valued non-terminal nodes.

Operators	Operands
=, ≠, <, >, ≥, ≤	Query or Node value (multi-valued)
contains	Element(Query)
exists, forall, none	Query

Table 2. Operators and operands for a multi-valued non-terminal node

4.2 The querying interface

The WDBQL querying interface is proposed via an HTML page. It contains three distinct interfaces which are the navigation, the contextual filter and the summary interfaces. The navigation interface allows the user to consult the database schema in accordance with the navigational process described before. This interface supports the user model. The contextual filter interface is used to define the result of the query and to express selections. It is a guide to the casual user as it saves him from making inconsistent selections for one's query. Finally, the summary interface reminds the user the expressed query. We give the figure 5 as an example of the querying interface.

The expressed query looks for all the boroughs having a young mayor, i.e. a mayor who is less than 35 years old. The navigation interface[1] shows the navigation from *Borough* to *Mayor*, and from *Mayor* to *Age*. The contextual filter interface[2] illustrates the expression of a selection. Operators and operands must be specified to express a predicate of selections. The user chooses an operator in the Operators contextual list. This list proposes all the operators available for the selected node (here *Age*). Then, the user enters the value of the operand. Finally, the user chooses in the Action/predicates list if he wants to execute the query, abort it, or go on the predicate specification using And or Or logical connectors. The summary interface[3] reminds the user the expressed query: the result is *Borough* and the selection criteria is *Age<=35*.

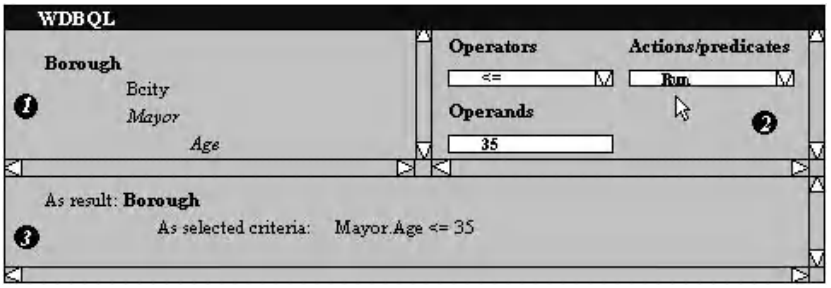


Fig. 5. The WDBQL interface

4.3 Comparison with related works

Recent development mixing Web and database technologies are numerous. We distinguish three classes of work related to information management on the Web: (i) modelling and querying the Web, (ii) information extraction and integration, and (iii) Web site construction and restructuring. Even if WDBQS mixes Web and database technologies, it has a quite different objective which is to define a Web-interface for querying distant RDB and OODB. WDBQS is mainly intended for casual users. The language it proposes, called WDBQL is comparable to the different visual query languages described in the literature. These query languages may be classified into form-based, graph-based, icon-based and Web-based query languages.

Form-based query languages allow the expression of queries from forms which represent the queried database schema. Form-based query languages are proposed for RDB (QBE [18], Access [1]), for extended-RDB (VQL [17], QBEN [13]) and for OODB (OOQBE [16], GOQL [12], PESTO [7], RYBE [13]). Graph-based query languages permit the expression of queries from graphs which describe the queried database schema. Some graph-based languages adopt a conceptual approach (QBD [3], CONQUER [6]) while others adopt an object-oriented one (OHQL [2], QUIVER [10]). Icon-based query languages use visual metaphors to both represent the queried database and the query language operators. ICI [8] and VISTA [4] are examples of icon-based query languages. Web-based query languages permit the expression of Web queries i.e. queries expressed from a Web interface using a navigational language. DataGuides [11] and WDBQL are examples of Web-based query languages. DataGuides is dedicated to semi-structured databases while WDBQL is dedicated to remote RDB and OODB. WDBQL is ODMG-compliant.

5 Conclusion and future works

This paper has presented WDBQS, a system we developed to allow LC casual users to easily query with a common language distant RDBs and OODBs. WDBQS has a client-server architecture ; it is accessible via the Web in an intranet context. WDBQS

proposes WDBQL, a query language well-suited for LC casual users. WDBQL is a Web-based query languages. That means it advocates a Web-like query formulation mechanism. Such a mechanism is really easy to use for casual users.

If we compare WDBQS to other similar systems [6, 7, 11, 15], we can mention some of the main differences:

- WDBQS allows to query with a common language distant RDBs and OODBs. So, it is different from [11] which is dedicated to semi-structured databases.
- WDBQS is ODMG-compliant.
- WDBQS proposes a Web-based query language i.e. a language which is accessible via the Web and which advocates Web navigational mechanisms for query building. So, it is different from [6] and [7] which propose a navigational query language but which do not have a Web-interface.

However, WDBQS has some drawbacks. On the one hand, WDBQL is less powerful than OQL. It needs to be extended in order to include (i) nested queries -we have implemented an operator whose name is NewQuery and it is possible to nest a query using the in operator; predicate such as x in (Query) are available -, (ii) aggregation operators -Count, Max, Min, Avg and Sum are not yet available- and (iii) collection operators -Flatten, Element, Union, Intersect and Minus are neither available-. These operators need to be integrated to WDBQL in order to make it powerful and comparable to OQL. But, we are not sure we should implement them because we keep in mind that WDBQL is only intended for Local Communities casual users who express enough simple queries only including selection, projection or join operations. The user-friendly objective that WDBQL has for this kind of users must not be given up to the benefit of its power. On the other hand, we need to set evaluations of WDBQL up to LC casual users.

WDBQS is still an ongoing project. A first prototype is now available. Only the Oracle RDB and the O2 OODB are accessible as remote databases. We implemented two wrappers which respectively translate Oracle and O2 database schemas as instances of the meta-schema. We used OracleWeb and O2Web to display the WDBQL query results. We now implement the wrapper whose aim is to translate a query result in a remote database format into the WDBQL format. We also implement Jasmine as a remote database for WDBQS.

References

1. Access: Microsoft home page: http://home.Microsoft.com
2. Andonoff, E., Mendiboure, C., Morin, C., Rougier, V., Zurfluh, G.: OHQL: An Hypertext Approach for Manipulating Object-Oriented Database. Information Processing and Management, Vol. 28, n°6, 1992
3. Angellacio, M., Catarci, T., Santucci, G.: Query By Diagram: a Fully Visual Query System. Visual Languages and Computing, Vol. 1, n° 2, 1990
4. Belieres, B., Trepied, C.: New Metaphors for a Visual Query Language. DEXA'96, International Workshop on Databases and Expert Systems Applications, Zurich, Switzerland, September 1996

5. Belloir, N.: Une approche navigationnelle pour l'interrogation de base de données via le Web. Mémoire de DEA, Institut de Recherche en Informatique de Toulouse, Toulouse, France, September 1999
6. Bloesch, A., Halpin, T.: CONQUER: a Conceptual Query Language. ER'96, 15th International Conference on the EntityRelationship Approach, Cottbus, Germany, October 1996
7. Carey, M., Haas, L., Maganty, V., Williams, J.: PESTO: An Integrated Query/Browser for Object Database. VLDB'96, 22nd International Conference on Large Data Base, Mumbai, India, August 1996
8. Catarci, T., Massari, A., Santucci, G.: Iconic and Diagrammatic Interfaces: an Integrated Approach. International Workshop on Visual Languages. Kobe, Japan, June 1991
9. Cattell, R.G.: ODMG-93: Le Standard des Bases de Données Objet. International Thomson Publishing, France, 1995
10. Chavda, M., Wood, P.: Towards an ODMG Compliant Visual Object Query Language. VLDB'97, 23rd International Conference on Very Large DataBase, Athens, Greece, August 1997
11. Goldman, R., Widom, J.: DATAGUIDES: Enabling Query Formulation and Optimisation in Semi-structured Databases. VLDB'97, 23rd International Conference on Very Large DataBase, Athens, Greece, August 1997
12. Keramopoulos, E., Pouyioutas, P., Sadler, C.: GOQL: a Graphical Query Language for Object-Oriented Database Systems. BIWIT'97, 3rd Basque International Workshop on Information Technology, Biarritz, France, July 1997
13. Lorentzos, N.A., Dondos, K.A.: Query By Example for Nested Tables. DEXA'98, International Workshop on Databases and Expert Systems Applications, Vienna, Austria, August 1998
14. Sallaberry, C., Bessagnet, M.N., Kriaa, H.: RYBE: a tabular interface for querying a multidatabase system. COMAD'97, 8th International Conference on Management of Data, Madras, India. December 1997
15. Sivadas, M., Fernandez, G.,: GeM and WeBUSE: Towards a WWW-Database Interface. Workshop on Co-ordination Technology for Collaborative Applications, Asian'96, 2nd Asian Computer Science Conference, Singapore, 1996
16. Staens, F., Tarentino, L., Tiems, A.: A Graphical Query Language for Object-Oriented Databases. International Workshop on Visual Languages, Kobe, Japan, June 1991
17. Vadaparty, K., Aslandignan, Y.A., Ozsoyoglu, G.: Towards a Unified Visual Database Access. SIGMOD'93, 20th ACM SIGMOD International Conference On Management of Data, Washington, USA, 1993
18. Zloof, M.: Query By Example: A Database Language. IBM System Journal, Vol. 16, n°4, 1977

Performance Issues of a Web Database

Yi Li, Kevin Lü

School of Computing, Information Systems and Mathematics
South Bank University
103 Borough Road, London SE1 0AA
{liy, lukj}@sbu.ac.uk

Abstract. Web databases are becoming an efficient tool used to manage the web sites. In this paper we analyse the performance of a typical Web database system with different sizes of web pages and different sizes of database tables. Since a web server and a database server work simultaneously, the response time in dealing with a request to the database can not be seen simply as the web server service time plus database service time. The performance metrics and optimisation suggestions are made on the basis of the analysis of the relationship between them. Initial experiments are designed to investigate how a Web database system works and what affects its performance, in particular, the response time. We explored the different ways of sending query result files. An analysis of the initial test results and suggestions on improving the Web database system performance are presented.

1 Introduction

The World Wide Web (WWW) has developed into the largest information database and, potentially, the most convenient medium for businesses. Web sites used for conducting business transactions have, in fact, become one of the most promising factors that determine the success or failure of an enterprise [1]. Good performance of Web databases provides a company with a definite edge over competition while poor performance makes it seriously handicapped.

Hence to ensure good performance of Web databases is absolutely essential for business institutions as well as for any type of enterprises. A Web database can be seen as an integrated system of a web server and database servers. If data files are stored in web databases, then queries to the databases normally take longer time than requests to static Web pages or Web site script pages. Therefore they directly determine the Web server performance [2]. It is very important to understand how Web databases work and to make them work with maximum efficiency.

The objectives of this study are: to investigate through experiments the way a Web database system works; to find performance matrices to accurately describe the Web database system; to find ways of tuning the system to get better performance; and to estimate the performance of the system. In this paper we present initial experiments on a Web database system performance and an analysis of the test results. Relationships between query result file size, paged result size, database base table size, network throttle, and the response time are examined.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 825–834, 2000.

The remainder of the paper is organised as follows: Section 2, discussion of the related work. Section 3, characteristics of the web database system to be investigated. Section 4, experiments and results. Section 5, analysis of the results of the experiments and the web database model, and Section 6, conclusions and suggestions for further research.

2 Related Work

The database research community has an immense interest in the management of Web information. A series of conferences on Web databases (e.g. International Workshop on the Web and Databases) have been held since 1998 [3]. The demand for extending the functions of databases to the Web has presented new challenges to database researchers and developers as can be seen in some proposals relating the database area for data management on the Web sites [2].

Most recent efforts of Web database developers are directed towards exploring the specifications of the structure and content of Web sites on how to get useful information quickly from Web databases. Efforts have been made on modelling the Web as an infinite, semi-structured set of objects and providing a set of language for managing and restructuring data coming from the Web [13].

In relation to measurement of the capacity of Web sites, several benchmarks have been advanced by different institutions. The first published of these was the SGI Webstone benchmark [4]. SPECweb99, a standardised benchmark, was developed by the Standard Performance Evaluation Corporation (SPEC) to measure the maximum number of simultaneous connections. It asks for a predefined benchmark workload that a Web server is able to support while still meeting specific throughput and error rate requirements [5]. TPC-W is a new Web application benchmark recently launched by the Transaction Processing Performance Council (TPC) to measure the performance and price performance of computer systems used for transactional Web environment [6].

There are no suitable performance metrics and benchmarks for measuring the Web database system yet. Current performance studies either focus on the Web server or on the database server. We think there is a need to conduct a performance study taking the Web database as an integrated system.

3 Architecture of a Web Database System

A Web database is an integrated system of Web severs and database servers, which enables users to access on-line information in a platform-independent manner through Web browsers.

Web servers and database servers work together in a Web database as an integrated system. Typical Web databases are three-tier or N-tier [7]. The users run standard browsers on the client side and the requests are transferred over the Internet to Web database systems. A Web server can create a number of threads and each thread serves a request from a client.

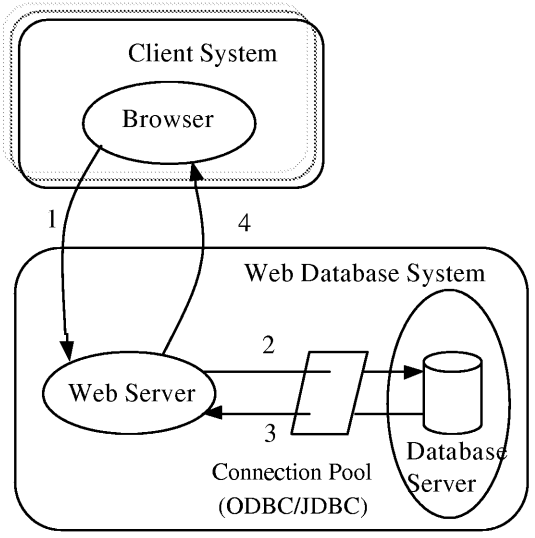


Fig. 1. Processing a Web database query at a typical Web database system

Figure 1 shows a typical Web database processing procedure. The following outline the details of a query processing in a Web database system.

Step 1. Query requests are sent from clients to a Web database system via Internet.

Step 2. Each request is served by a thread called initialisation. Then the Web server deals with the request. When the Web server finds a database query, it will dispatch the query request to the database server through an interface (the connection pool in Figure 1).

Step 3. A query result made by the database server is given back to the Web server through the same interface.

Step 4. The Web Server incorporates the query result into a response Web page, and then sends the page back to the client.

The Web server and the database server in a Web database system may work concurrently. The two servers can run either on the same computer or on two different computers. From the clients' point of view, they are one integrated system, which acts as a Web database. Web servers play the role of a service provider and data consumer, responsible for interacting with clients and requesting data from a database on their behalf. The database server acts as a data provider in charge of data storage and data manipulation.

Pooling technology is used in most Web database systems to get better performance. There are usually two different types of pools. One is a thread pool in the Web server and the other is a connection pool to the database server.

Thread pools eliminate the overheat of creating a new thread when an additional request reaches the server. Instead of creating a new thread for each request, the server creates a pool of threads when it starts up. If more CPU processing power is available, then by increasing the number of threads in the pool the server capacity will be improved because an available CPU processing power can be used to deal with more requests concurrently. But too many threads will consume a lot of resources and will weaken the server performance.

The same mechanism is applied in the connection pool. At the early stage of the Web database development, Common Gateway Interface (CGI) was the only way used to connect a Web server to a database. The CGI architecture, which gave poor performance, was gradually replaced by the Application Programming Interface (API) approach [8], in which an interface between the server and back-end applications is created by using a dynamic linking or shared object. The ODBC pool is a typical connection pool. Vendors of database systems integrate databases with the Web server through their own API to act as an extension to the Web server that can communicate with databases through ODBC [9].

4 Experiments

A series of tests are designed to examine how a Web server and database servers work together and what may affect the response time of a Web database system. The following relationships are examined namely, relationships between: 1) query result file sizes and response time; 2) table sizes of a database and response time; 3) the network throttle and response time; and 4) paged query result file sizes and response time.

4.1 Experimental Design

The tests we carried out simulate the activities of a Web database system and clients' Web browsers. In our experiments, Windows NT is used as the platform with Microsoft Internet Information Server as the Web server, MS SQL Server as the database server, and the Active Server Pages as the Web site script language. A thread pool initialises new requests and an ODBC pool is connected to the database server.

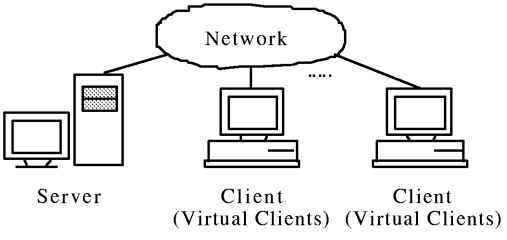


Fig. 2. Components for the tests

The tests involve three primary components: a Server, Clients, and a Network. During the tests, a client runs a virtual client program and the server responds with a result file. Figure 2 shows these components of our tests.

A Web server and a database server are running on the same Server computer. The Server uses a set of prepared sample ASP pages and sample databases, which simulate those that a server might provide for its clients. The prepared ASP pages and databases simulate Web databases of various sizes.

A client application is running on a computer to simulate one/many client's browser(s) and virtual clients. The client application can be run on more than one client computer. Each virtual client makes one connection and a page request to the server at a time. This design enables each client computer to simulate more than one client.

In these tests, the network refers to the communication links between the client computers, and the server computer. It is a 10Mbps Ethernet network connected by a hub.

4.2 Performance Metrics

Traditionally, response time and throughput are the two most important performance metrics for both Web information systems and database systems [11][12]. But the Web database has its unique character that is different from any other database applications. Suppose the result size of a Web database query is 1MB, a client may only need to receive the first page to read at first, expecting pages to follow will be available by the time when he finishes reading the first page. The client does not mind whether the whole result file arrives at the same time or in sequence. If the client can receive the sub-sequential pages just before completing the page in hand, he will feel being served effectively, and would be satisfied with the service. Meanwhile, clients do not want the last page to arrive too late. Therefore, we adopt the following two metrics as our response time metric:

TTFB - time for a client to receive the first response byte from the server.

TTLB - time for a client to receive the last response byte from the server.

In addition to the above two kinds of response time, the following new factors need to be considered to get a clearer description of the performance of a Web database system.

Result Arrival Rate (RAR) is used to present the time intervals between the two consecutive pages. RAR is measured in terms of bits per second (bps) or pages per second (pps). A reasonable RAR can improve the system resource utilisation without degrading the service quality provided for users.

Throughput Ratio (TR) indicates how the capacity of the Web server matches the capacity of the database server in a Web database system. The throughput of a Web server and the throughput of a database server both refer to the amount of data they can process in a certain time. If their throughput ratio is nearly equal to 1, i.e., their processing speeds match each other, and the Web database system can achieve its maximum throughput.

4.3 Results

A set of tables (10,000, 20,000 and 60,000 records) were created for the tests, and the length of a record ranged from 86 bytes to 256 bytes. So, there were totally 6 tables used in our tests. Queries to the database tables were adopted according to the standard SQL.

In order to control cache effect on the performance, tests carried out were of two types. One is called *single query* tests, in which in order to reduce the cache effect to a minimum we tested the queries one at a time once the Server was started up. The other is called *mixed* tests, in which we tested a request in a simulated environment.

Response time vs. query result file size. A number of queries were tested to get the response time of different result file sizes. Figure 3 shows the Web database system response time versus the result file size from 10k to 100k in *single* query tests.

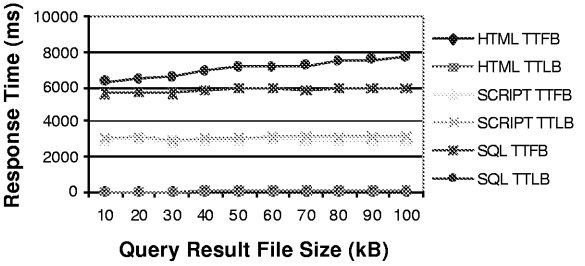


Fig. 3. Web database system response time vs. query result file size (*10k-100k single query tests*)

Our tests show that when the result file size increases the response time for queries to the database increases about 7.8 times more than the script response time, and the rate of increase of the script response time is about 1.5 times bigger than that of HTML as shown in figure 3. Therefore, users may ignore the small increases in response time of the static HTML when its result file size increases, but they can not ignore the increase in the response time of a Web database system when its result file size increases.

Based on our experiments, we found that the response time of HTML request is the shortest and the response time of database request is the longest among the three classified requests when the query result file sizes are the same. Figure 3 shows two other features: 1) Linear Relationship between Response Time and Query Result File Size. 2) The increase rate of database query response time is much higher than that of other requests in relation to the increase of the query result file size.

Our other test results confirm that the linear relationship that exists before the result file size is increased up to 1000kB under a *mixed* non-queueing environment.

The response time of a script request may be close to an HTML request because the script is running inside a Web server and can use API directly to improve the performance. The response time of a database query is much longer than that of an HTML request, because not only the query running in the database needs a relatively longer time, setting up a connection to the database and the communication of the query result data between the Web server and the database server also need time. When the response file size increases, the Web server time increases and so does the database server time. Further more, the communication time increases too. Therefore, the increase rate of database query response time is much higher than that of other requests in relation to response file size increases.

Response time vs. database table size. To see how much the database size affects the response time, we changed the database size from 10,000 records to 650,000 records in the tests. Figure 4 shows the testing result with the same requests to a different size database. A large database has a bit longer response time than a small one but the difference is small. The response time increase rate of a large database with the result file size is almost the same as that of a small database.

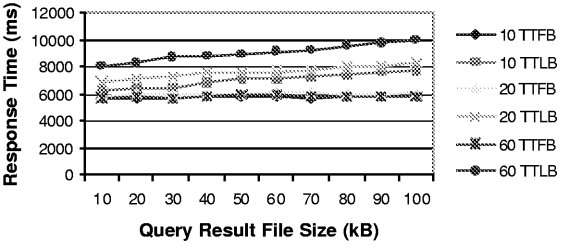


Fig. 4. Web database system response time vs. query result file size (*10k–100k 10k, 20k&60k records single query tests*)

Changing database size means changing database service time. The linear relationship between response time and query result file size remains. Figure 4 shows that the linear relationship is unchanged and the values are not increased significantly either. This is because a database server is efficient in querying data and the transferring time of query result file between the database server and the Web server is almost unchanged because of the same size of query result file. Therefore, the increase in database service time by data added to the database is small. If a query is very expensive to operate, an obvious increase in Web database system response time will occur when the size of a database increases.

Response time vs. network throttle. The Web server may work in two modes. In Mode 1 the data from database are processed and sent to clients one by one, on a first come first served basis. In Mode 2, only when all the data arrive are they processed in the Web server and then, sent to clients altogether. All the above tests were performed by means of Mode 1. There produced a slightly different result when we changed to using Mode 2, and the network transfer rate was limited to T1, which is 1.544Mbps to connect to the network.

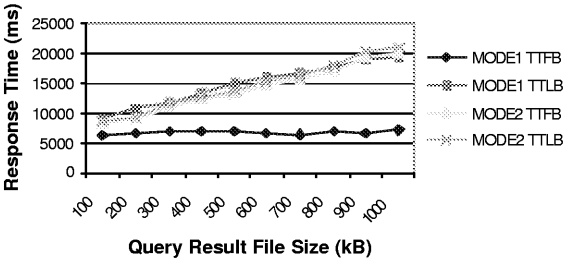


Fig. 5. Web database system response time vs. query result file size (*100k - 1000k Mode 1&2*)

Figure 5 shows that Mode 2 response time is normally shorter than Mode 1's. When the response file size is increased to a certain value, Mode 2 response time is longer than Mode 1 and the bandwidth of connection to the network becomes the bottleneck device.

The reason that Mode 2 line exceeds Mode 1 line is that when the result data get to a certain value, the data can not be sent out to the network immediately and queueing occurs. We do not recommend the use of Mode 2 when the response file size is big, because the queueing time will be counted.

Mode 1 is not the best way to gain good performance either, because its TTLB is longer than Mode 2 normally. It is better to send out the response data whenever they accumulate to a certain amount. Thus, it can control the queueing time by insuring that the server send the data out immediately, and it can also shorten TTLB by preventing the server from switching operations too frequently.

Response time vs. query result file paged size. When a result file is over 3 or 4 pages, it is better to divide it into several small pages (normally 1 or 2 pages to display) and send them to the client separately. We tested the response time of different size pages.

We defined 256 bytes for each record in a result page. The 62.5kB file was divided into pages of different records ranging from 8 to 15 records. Figure 6 shows the results.

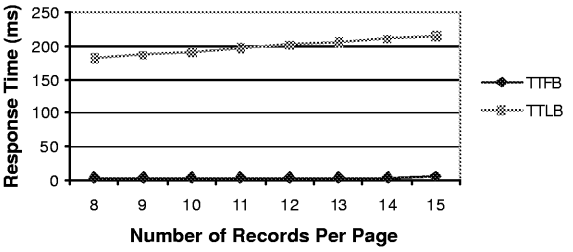


Fig. 6. Web database system response time vs. different numbers of records per page (*8-15 records mixed tests*)

If a result page with 8 records is changed to one with 15 records, the response time will increase about 18% according to Figure 8. We assume that clients do not care much about whether it is 8 or 15 records on the first result page they read. Then, using 8 rather than 15 records page the result file will have much better performance for the system (over 15% response time reduced).

From the above tests, we have come to the following findings.

1. The test results show that the time of response of a Web database is far more sensitive to the response file size. That is, if the response file size increases, then the response time for the Web database query will increase much more than the response time of the static Web pages or the Web site script pages.
2. Clearly, the response file size has immediate effects on the maximum permitting query rate [10]. Therefore, the rate of queries to the Web database is the main factor that determines the capacity of the whole system.
3. The response file size being so important to the performance of the system, it should be divided into separate pages to be sent to the clients separately and make the size in each page small within the limit of satisfactory service to the clients.
4. Because the time between TTFB and TTLB of database queries is much longer than that of static pages, metric RAR can be used to show clearly the service quality of the system. And if we know the system's TA, we can adjust the ratio of HTML, script and database pages according to workload on the system in order to get the best match between the Web server and the database server.

Conclusions

Several attributes related to the performance of Web databases are investigated from a perspective of taking Web servers and database servers as an integrated system. The importance of performance study of Web database systems cannot be over emphasised because queries to the database system are dealt with over much longer response time compared with queries to the script or HTML files.

New metrics for Web database systems are suggested to evaluate the system. They are: time for the first response byte from the server (TTFB), time for the last response byte from the server (TTLB), Result Arrival Rate (RAR), and Throughput Ratio (TR). These metrics are essential to accurately describe and evaluate a Web database system.

Our tests confirm that the way to send query results back to clients is essential to good performance of the system. To gain better performance the technology of paged query result file should be used for queries with a very large result file. The result page size is a key factor contributing to better performance.

We are currently working on experiments, which multiple users access to a Web database system at the same time with different arrival rate and different types of queries. The study aims to find a way to predicate and tune the performance of a Web database.

References

1. Daniela Florescu, Alon L. and Dan S.: Optimization of Run-time Management of Data Intensive Web Sites. Proceedings of the 25th VLDB Conference. Edinburgh, Scotland, (1999)
2. Daniel A. Menasce and Virgilio A.F. Almeida: Capacity Planning for Web Performance, Metrics, Models, and Methods. Prentice Hall, (1999)
3. Goldman, R., Mchugh, J., and Widom: Proceedings of the 2nd International Workshop on the Web and Databases. Philadelphia, Pennsylvania, June (1999)
4. Blakeley, Michael: WebStone FAQ. Silicon Graphics, Inc. 9 Nov (1995)
5. SPECweb99 Release 1.01 Specification: http://www.sprc.org/osg/web99/ November 8, (1999)
6. TPC BENCHMARK™ W (Web Commerce) Draft Specification: http://www.tpc.org/, November 19, (1999)
7. Robert Orfali, Dan Harkey and Jeri Edwards: Client/Server Survival Guide. Wiley (1999)
8. Ron Vetter: Web-Based Enterprise Computing. Computer, May (1999)
9. Leland Ahlbeck: Improving the Performance of Data Access Components with IIS 4.0. http://msdn.microsoft.com/workshop/server/components/daciiisperf.asp
10. Louis P. Slothouber, Ph.D, StarNine Technologies.: http://louvx.biap.com/white-papers/performance/overview.html
11. Lü K. J, Dempster E. W., Tomov N., Williams M. H.: Verifying a Performance Estimator for Parallel DBMS. Proceedings of Eura-Par 98, LNCS 1470. Springer (1998)
12. Lü K. J, Dempster E W, Tomov N T, Taylor H, William M H, Pua C S, Burger A, and Broughton P.: An Analytical Tool for Predicting the Performance of Parallel Relational Databases. Journal of Concurrency: Practice and Experience. John Wiley & Sons (1999) 1-16
13. Florescu, D.: Database Techniques for the Wide Web: A Survey. SIGMOD Record, 27(3), (1998) 59-74

Improving the Performance of High-Energy Physics Analysis through Bitmap Indices

Kurt Stockinger[1,2], Dirk Duellmann[1], Wolfgang Hoschek[1,3], and Erich Schikuta[2]

[1] CERN IT Division, European Organization for Nuclear Research
CH-1211 Geneva, Switzerland
{Kurt.Stockinger, Dirk.Duellmann, Wolfgang.Hoschek}@cern.ch
[2] Institute for Computer Science and Business Informatics
University of Vienna, A-1010 Vienna, Austria
Erich.Schikuta@univie.ac.at
[3] Institute of Applied Computer Science
University of Linz, A-4040 Linz, Austria

Abstract. Bitmap indices are popular multi-dimensional data structures for accessing read-mostly data such as data warehouse (DW) applications, decision support systems (DSS) and on-line analytical processing (OLAP). One of their main strengths is that they provide good performance characteristics for complex adhoc queries and an efficient combination of multiple index dimensions in one query. Considerable research work has been done in the area of finite (and low) attribute cardinalities. However, additional complexity is imposed on the design of bitmap indices for high cardinality or even non-discrete attributes, where different optimisation techniques than the ones proposed so far have to be applied.
In this paper we discuss the design and implementation of bitmap indices for High-Energy Physics (HEP) analysis, where the potential search space consists of hundreds of independent dimensions. A single HEP query typically covers 10 to 100 dimensions out of the whole search space. In this context we evaluated two different bitmap encoding techniques, namely equality encoding and range encoding. For both methods the number of bit slices (or bitmap vectors) per attribute is a central optimisation parameter. The paper presents some (first) results for choosing the optimal number of bit slices for multi-dimensional indices with attributes of different value distribution and query selectivity. We believe that this discussion is not only applicable to HEP but also to DW, DSS and OLAP type problems in general.

1 Introduction

One of the big challenges at CERN (the European Organization for Nuclear Research in Geneva, Switzerland) is the management of the large amount of data and the complexity of objects that are the results of the HEP experiments. In particular, sub-atomic particles are accelerated to nearly the speed of light

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 835–845, 2000.

and then collided. Such collisions are called *events* and are measured at time intervals of only 25 nanoseconds for some of the new experiments. According to [8] around 5 Petabyte of data will be written per year that will be analysed by some 5,000 physicists around the world over a life span of two to three decades.

Currently physics analysis tasks are based on sequentially scanning the preselected event space, obviously not very efficient for queries with small selectivities. In this case the usage of a proper multi-dimensional index data structure accelerates these processes by orders of magnitude.

In the literature on multi-dimensional access methods a variety of indices are proposed [5] ranging from spatial data access methods like the R-tree [6] or the BV-tree [4] and its variants to non-spatial data access methods like the Pyramid-tree [1]. However, all these indices are optimised for transaction processing, i.e. inserts, updates, deletes, etc. what is not the major need of HEP analysis.

Similar to DW applications and DSSs, HEP data are read-mostly and the access methods are characterised by multi-dimensional, highly complex queries. What is more, most of the queries are so called *partial range queries* where only a small subset of the whole search space is accessed. Multi-dimensional access methods like the Pyramid-tree show their best performance characteristics for *full range queries* and are thus only sub-optimal for *partial range queries*.

We therefore propose to use bitmap indices, which are optimised for processing complex adhoc queries in read-mostly environments. The basic idea of a bitmap index is to store one vector of bits per distinct attribute value (e.g. possible attribute values are *colours*). Each bit of the value is mapped to a record. The associated bit is set if and only if the record's value fulfils the property in focus (e.g. the respective value of the record is equal to *red*). [2] [3] [14] studied different kinds of bitmap encoding techniques but only for discrete values. However, additional complexity is imposed on the design and implementation of bitmap indices for non-discrete values since different optimisation techniques to the ones proposed so far have to be applied.

In this paper we make the following contributions to the research on bitmap indices:

- Applying bitmap indices to high performance physics experiments.
  We give a proof of concept that traditional physics analysis can be considerably improved by bitmap indices (BMIs) and show that this technique is very efficient for HEP-specific data distributions. What is more, we introduce *partitioned equality encoding*, which is a variant of equality encoded bitmap indices as used in [12]. In this case one of the most crucial points is the choice of the correct number of bins which highly depends on the number of indexed attributes, i.e. the number of dimensions of our search space.
- Employment and analysis of BMIs on top of an object database managment system (ODBMS).
  In contrast to [12] where bitmap indices are implemented on top of a mass storage system, we present a first implementation on top of an ODBMS, namely Objectivity/DB [9] . In addition, we also provide an extensive performance analysis and thus characterise the features of bitmap indices for

HEP, which can be, without loss of generality, directly applied for any read-mostly environment like DWs and DSSs.
- Studying the impact of non-discrete data values.
  Finally, we discuss the impact of *partitioned range encoding*, which is a new variant of range encoded bitmap indices [2] and is not covered in the research community so far.

The paper proceeds as follows. In Section 2 we give a survey of related work and outline the differences to our approach. In Section 3 and 4 we discuss BMIs for HEP analysis and how we implemented them on top of an ODBMS. A detailed evaluation of our index is presented in Section 5 where we elaborate on the optimal binning of the BMI and apply these results to data distributions which are very common in HEP analysis. In Section 6 we discuss the impact of range encoded BMIs and finally conclude our work in Section 7.

2 Survey and Discussion of Related Work

A detailed discussion on designing bitmap indices based on different encoding schemes is presented in [2] and [3]. In particular, space and time complexities for so-called equality encoded, range encoded and interval encoded bitmap indices are evaluated. Equality encoding (Table. 1 (b)) can be regarded as the most fundamental method that consists of $|A|$ bitmaps (bitmap vectors) where $|A|$ is the cardinality of the attribute to be indexed on. This type of index is optimal for exact match queries of the form $Q_e : v = a_i$. One sided-range queries like $Q_{1r} : v_1 \text{op } a_i$ where $op \in \{<, \leq, >, \geq\}$ show the best performance characteristics with range encoded bitmap indices (Table 2 (b)), which only consist of $|A| - 1$ bitmap vectors. Finally, interval encoding (Table 2 (c)) consists of $\frac{|A|}{2}$ bitmap vectors only and is optimal for two-sided range queries $Q_{2r} : v_1 \text{op } a_i \text{ op } v_2$ where $op \in \{<, \leq, >, \geq\}$. All the optimisation methods presented in their work address discrete attribute values only. However, since our data are contiguous and thus do not have finite cardinalities, different optimisation techniques than the ones proposed so far have to be applied.

Table 1 and 2 depict these different encoding techniques for the same set of attribute values. According to the terminology of [10] [11] the values in Table 1 (a) are referred to as *projection index* whereas the other methods are called *bit sliced* indices.

One of the major problems of simple bitmap indices, namely handling of large cardinality domains, is solved in [15] by range-based indices. A bitmap vector is used to represent a range instead of a distinct value and the entire ranges are partitioned into equally spaced *buckets*. As is the case with the approach described in this paper, range-based indices require additional query processing time to examine the details of all the records in the matched buckets. However, a detailed analysis and a possible solution to the problem of the additional overhead for retrieving data from disk ("sieving out" the matching attribute values), was still left an open issue.

Table 1. a) Projection Index π_A and b) Equality Encoding E^i

	π_A(R)	E^9	E^8	E^7	E^6	E^5	E^4	E^3	E^2	E^1	E^0
1	4	0	0	0	0	0	1	0	0	0	0
2	3	0	0	0	0	0	0	1	0	0	0
3	4	0	0	0	0	0	1	0	0	0	0
4	5	0	0	0	0	1	0	0	0	0	0
5	7	0	0	1	0	0	0	0	0	0	0
6	8	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	0	1
9	4	0	0	0	0	0	1	0	0	0	0
10	8	0	1	0	0	0	0	0	0	0	0
11	6	0	0	0	1	0	0	0	0	0	0
12	7	0	0	1	0	0	0	0	0	0	0

Table 2. a) Projection Index π_A, b) Range Encoding R^i and c) Interval Encoding I^i

	π_A(R)	R^8	R^7	R^6	R^5	R^4	R^3	R^2	R^1	R^0	I^4	I^3	I^2	I^1	I^0
1	4	1	1	1	1	1	0	0	0	0	1	1	1	1	1
2	3	1	1	1	1	1	1	0	0	0	0	1	1	1	1
3	4	1	1	1	1	1	0	0	0	0	1	1	1	1	1
4	5	1	1	1	1	0	0	0	0	0	1	1	1	1	0
5	7	1	1	0	0	0	0	0	0	0	1	1	0	0	0
6	8	1	0	0	0	0	0	0	0	0	1	0	0	0	0
7	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1
8	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1
9	4	1	1	1	1	1	0	0	0	0	1	1	1	1	1
10	8	1	0	0	0	0	0	0	0	0	1	0	0	0	0
11	6	1	1	1	0	0	0	0	0	0	1	1	1	0	0
12	7	1	1	0	0	0	0	0	0	0	1	1	0	0	0

Another encoding technique based on binary encoding is proposed by [13] where an attribute value is represented in binary form with only $\lceil log_2 |A| \rceil$ bitmaps. Obviously, the storage overhead is much less for high cardinality attributes when compared to equality encoding or range encoding but even according to the authors, an optimal solution for evaluating the queries might not always exist.

Static and dynamic query optimisation for *continuous range selections* (i.e. one-sided and two-sided range queries) and *discrete range selections* (i.e. queries of the form $v \in a$ and $v \notin a$ are presented in [14]. Static query optimisations are questions concerning the optimal design of bitmaps and algorithms based on logical reductions. Dynamic query optimisation tries to answer questions on inclusion and exclusion for bit-sliced and encoded bitmap indices.

Currently the only work on BMIs for HEP is presented in [12]. Their work is based on a hybrid approach of equality encoded [2] and range-based BMIs [15] on top of a mass storage system. They also use bitmaps for their query optimiser to provide a quick estimate of the size of the requested data. However, an answer to the optimisation problem of the central optimisation parameters for designing a BMI for HEP, namely the optimal number of buckets (or bit slices), was not given yet.

3 Bitmap Indices for HEP

The typical query profile of physicists who wish to retrieve data for their analyses can be regarded as partial range queries, i.e. queries that do not cover all dimensions of the whole search space and thus only a subset of all dimensions of the data is retrieved. What is more, data are read-mostly and skewed.

In our prototype implementation we created a bitmap index for HEP data comprising 10^6 objects, i.e. 1 million events with up to 20 independent attributes. This can be regarded as an index table with a length of 10^6 and a width of 20. We assume that the order of the objects, that are stored in the index, does not change.

Similar to [12] we also use a hybrid approach of equality encoded [2] and range-based BMIs that we call *partitioned equality encoding* or short *equality encoding*. The properties or attributes are partitioned into bins, for example the attribute energy can be binned into several ranges like [0;20) GeV (Giga electron Volt), [20;40) GeV, etc. Afterwards, a bit slice is assigned to each bin, whereas *1* means that the value for the particular event falls into this bin and *0* otherwise.

The steps for performing a two-sided range query of the form $Q_{2r} : v_1 \text{op } a_i$ op v_2 where $op \in \{<, \leq, >, \geq\}$ are as follows. First, the query range has to be interpreted in terms of bins. Thus, we can easily compute how many bins need to be scanned for answering our query. Since each bin represents a range rather than a distinct value, the edge bins are so called "critical" bins, that might only be partially covered by the query condition. In order to "sieve" out the correct events from the *candidate slices*, we need to fetch the event data from disk and check the attribute value against the query condition. We refer to this as the "candidate check overhead" that makes the index highly I/O bound for a large number of candidates in the two candidate slices. Those slices that are covered 100% by the query range, are called *hit slices*. In this case all events that are represented by this slice are hits and do not need any additional checking. A typical example of a two-sided range query with 2 candidate slices and 1 hit slice is depicted in Fig. 1.

4 Implementation on Objectivity/DB

Basis for our implementation is Objectivity/DB, which is a distributed object database management system for high performance, high availability multi-tier applications. Objectivity/DB provides a robust, scalable multi-threaded

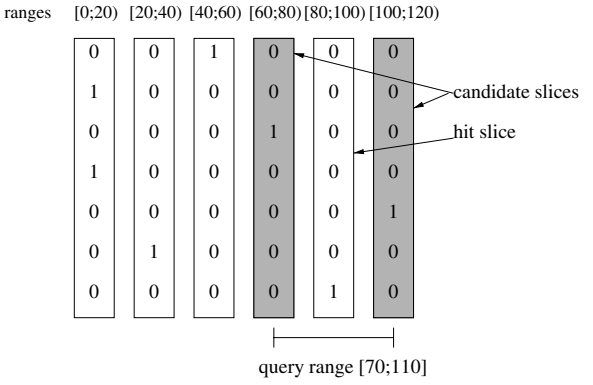


Fig. 1. Two sided-range query on a partitioned equality encoded HEP-BMI

database engine. Both the event data and the BMI are implemented in separate databases under one federation which in turn is the highest level of abstraction in Objectivity/DB and allows to be accessed physically distributed databases. From the point of view of the programmer, the whole database system is one logical unit. The main architectural aspects are depicted in Fig. 2.

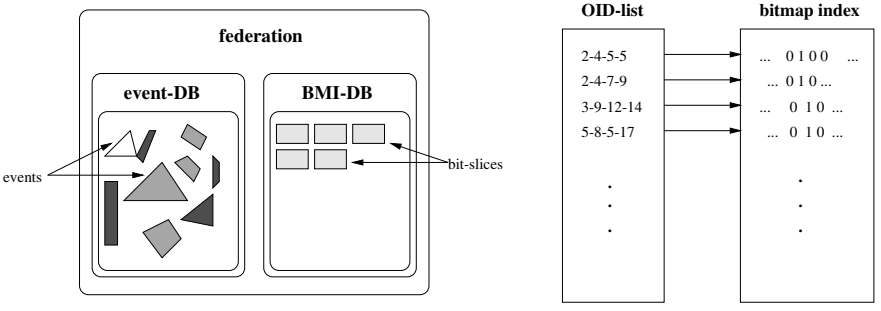


Fig. 2. Architectural overview of the BMI on top of Objectivity/DB

Any object in Objectivity/DB can be directly accessed by its object identifier (OID) which we use for keeping track of the event data. In particular, each physics event is stored as an object and can thus be directly accessed via its OID. This step is necessary, for example, for checking the *candidate slices*.

As we can see on the right side of Fig. 2, one OID-list is maintained in addition to the BMI. For instance, if we want to check the event at position x, we simply refer to the OID list at position x and fetch the event from disk for checking the attribute value against the query condition.

5 Justification of the BMI Approach

We carried out our benchmarks on a Pentium II 400 under Linux Red Hat 5.1. The BMI is implemented on top of Objectivity/DB version 5.1.2. Throughout the rest of this paper all experiments operate on 10^6 events.

5.1 Optimal Binning - Space/Time Complexity

The right number of bit slices (bins) can be regarded as one of the "key parameter" of this kind of bitmap index. A detailed discussion on the behaviour of the BMI with a different number of bins and a different number of indexed attributes is vital for the understanding of BMIs for HEP. To the best knowledge of the authors, this kind of investigation has not been done before for the special needs of HEP data and DW applications in general. The main motivation was a similar implementation of BMIs presented in [12] where 20 bins were chosen. However, no analysis or justification for this "key parameter" is given.

We will first elaborate on the optimal number of bins for queries against 1 indexed attribute and later extend our analysis onto 2, 10 and 20 attributes - that can be summarised as the "most characteristic" use cases of end-user physics analysis in HEP. For simplicity we base our analysis on uniformly distributed data since this gives us the best insight into the performance of the index. Again the domain selectivity $\sigma_{dom} = 100\%$.

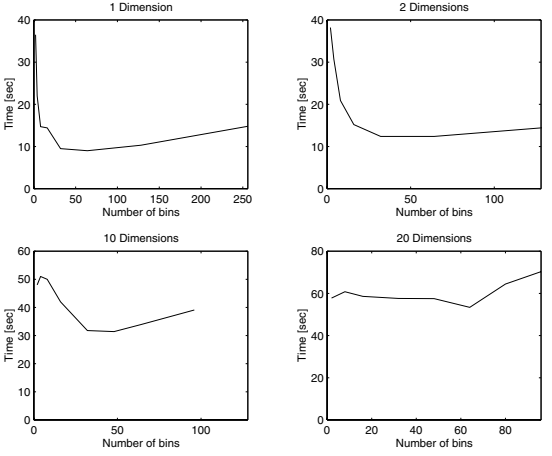


Fig. 3. Optimal binning for queries over various dimensions

As we can see from Fig.3, the optimal binning highly depends on the number of dimensions, that are covered by the range query. Generally speaking we can conclude that the higher the search space, that is covered by the query, the higher is the number of bins for an optimal query performance.

Let us first analyse the graph for queries over one dimension. As we might expect, the performance is worse for 2 bins since both of them are candidates and hence need to be checked against the query constraint. In this case, all events must be fetched from disk. By increasing the number of bins, the number of candidates in the candidate slices gets lower and thus the I/O spent on fetching events from disk is reduced. However, at the same time a larger number of bit slices has to be scanned for performing the Boolean operations on them. The optimal number of bins can be found at that point where these two effects offset each other.

The same effect can be found for higher dimensional queries with the slight difference that the optimum moves to the right, i.e. higher number of bins, due to the higher number of candidates in the candidate slices for a higher number of bins. Obviously, for 2 bins the number of candidates is the same for all dimensions but this number decreases more slowly as the number of dimensions and bins increases.

5.2 Different Distributions of Physics Data

After studying the behaviour of the BMI on event data following a uniform distribution, we will now motivate our implementation by applying it to typical data distributions that can be found in HEP. During this section we demonstrate the advantage of our index data structure for HEP analysis over conventional methods.

A great majority of physics data follows a Gauss or exponential distribution. Driven by the needs of physics analysis, we studied the behaviour of our BMI on a Gauss distribution with the parameters $\mu = 0$ and $\sigma = 1$, and on an exponential distribution with the parameter $\lambda = 1$. The number of bins is set to 32 and the number of dimensions covered by the query is 10. The results are summarised in Fig. 4

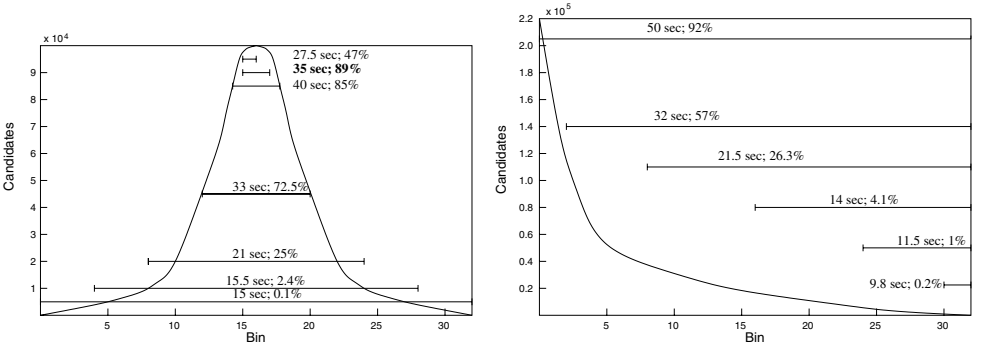


Fig. 4. Variable selectivities - Gauss and exponential distribution

In both figures a schematic view of the underlying data distribution is given. The horizontal bars indicate the domain selectivity of the query whereas the percentage on the bars refers to the selectivity of the candidate objects. In addition, the time for executing the query is given. As we can see in the figures the query time highly depends on the selectivity of the candidate objects and not on the domain selectivity.

The performance of the BMI is highest for the "lower end" of highly skewed data (e.g. the right most events that follow an exponential distribution). However, we also have to point out the poor performance of range queries at the "higher end" (worst case).

One possible optimisation for reducing the negative effects of the worst case would be a dynamic binning that adopts to the distribution of the event data. Since in most cases the access patterns of physics analysis are characterised by range queries around the "lower end" of the event data, our current implementation proofed to be the most promising approach for the HEP experiments at CERN.

6 Partitioned Range Encoding

Since one-sided range queries (Q_{1r}) are the most common kind of query in HEP, we also analysed range encoding [2] that performs considerably better than equality encoding. In contrast to [2] who based their optimisation techniques on discrete attribute values (where the problem of candidate and hit slices does not occur), we apply this method to contiguous values and thus a new optimisation method has to be considered. We refer to our approach as *partitioned range encoding* or in short *range encoding*.

The main advantage over equality encoded BMIs is that in the worst case only one bit slice has to be scanned for one-sided range queries per dimension (independent of the selectivity of the query). As for equality encoded bitmaps, all bit slices have to be scanned.

Since we have already studied the behaviour of equality encoded BMIs and raised the I/O problem of candidate slices, we can easily conclude from these observations on the impact of range encoding. As already mentioned, in the worst case one bit slice needs to be scanned for one-sided range queries per dimension. This also implies that we have to consider only one *candidate slice* and no *hit slice* at all.

Let us analyse the performance characteristics of one-sided range queries with both equality encoded and range encoded BMIs with variable selectivities, 10^6 objects and 10 indexed attributes. Again we studied the performance characteristics of the BMI on uniformly distributed data.

As we can see in Fig. 5, the query time for equality encoded BMIs increases with increasing selectivities (i.e. a higher number of *hit slices* has to be read) whereas the query time for range encoded BMIs is more or less constant for all selectivities (since no *hit slices* at all have to be scanned).

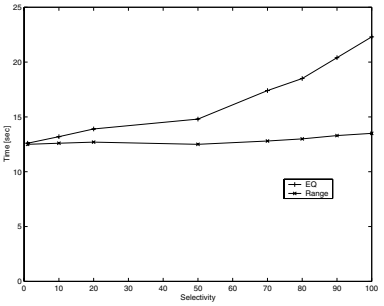


Fig. 5. Partitioned Equality Encoding (EQ) vs. Partitioned Range Encoding (Range)

The fact that only one bit slice needs to be scanned gives us much more freedom in extending the number of bins until the theoretical maximum of the cardinality of the attribute value. Since we are dealing with contiguous values, we do not have a finite cardinality and hence are mainly "restricted" by the space complexity. We thus end up in a "practical" optimisation problem, that is constraint by the available disk space.

Experimentally we can show that the behaviour of a range encoded BMI both for Q_{1r} and Q_{2r} corresponds to a partial scan over the event data. The performance characteristics are thus highly dependent on the characteristics of the underlying OODBMS, namely Objectivity/DB and the disk. In particular, for page selectivities between 5% and 100%, the read rate is almost linear. However, a significant speedup is achieved, if the page selectivity is smaller than 5% [7] which in turn is very common for multi-dimensional range queries in HEP analysis.

7 Conclusion

We have given performance studies of BMI for HEP data and pointed out the main difference to other studies on BMI - those that concentrated on only discrete attribute values. The main bottleneck has been shown to be the checking of the candidate slices due to the additional I/O for fetching the event data from disk in addition to the I/O for the BMI.

We have designed and implemented our BMI on top of a commercial object database management system, namely Objectivity/DB and used different bitmap encoding techniques and different data distributions for our analysis. As for *partitioned equality encoding* and uniformly distributed data we solved the "candidate-bottleneck" by increasing the number of bins and came to an optimal query performance. This optimum can be regarded as a trade-off between a high number of candidates and consequently more I/O on the event data vs. a low number of candidates and therefore a higher number of bins.

Since HEP queries are mainly one-sided range queries, we also studied *partitioned range encoding* where we discussed a completely new problem, namely

the behaviour of the BMI on attributes with infinite cardinality (as it is true for non-discrete values). We showed that the performance of range encoded BMIs clearly outperforms equality encoded BMI. However, there is no "optimal" number of bins, as is the case for equality encoded BMI. This gives the designer of BMI for HEP data a high degree of freedom since the number of bins (and indirectly the number of candidates) and thus the performance of the entire index is "only" restricted by the capacity of the storage space.

References

1. S. Berchtold, C. Boehm, H.-P. Kriegel, The Pyramid-Tree: Breaking the Curse of Dimensionality, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 1998, Seattle, Washington, USA
2. C. Chan, Y.E. Ioannidis, Bitmap Index Design and Evaluation, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 1998, Seattle, Washington, USA
3. C. Chan, Y.E. Ioannidis, An Efficient Bitmap Encoding Scheme for Selection Queries, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1999, Philadephia, Pennsylvania, USA
4. M. Freestone, A General Solution of the n-dimensional B-tree Problem, SIGMOD Record (ACM Special Interest Group on Management of Data 24(2), June 1995
5. V. Gaeda, O. Guenther, Multidimensional Access Methods, Computing Surveys 30, September 1998
6. A. Guttman, R-trees: A Dynamic Index Structure for Spatial Searching, Proc. ACM SIGMOD, Int. Conf. on Management of Data, 1984
7. K. Holtman, P. v. d. Stok, I. Willers, Automatic Reclustering Objects in Very Large Databases for High Energy Physics, Proceedings of IDEAS 1998, Cardiff, UK
8. http://www.cern.ch/MONARC/
9. http://www.objectivity.com/
10. P. O'Neil, Informix and Indexing Support for Data Warehouses, Database and Programming Design, February 1997
11. P. O'Neil, D. Quass, Improved Query Performance with Variant Indexes, Proceedings ACM SIGMOD International Conference on Management of Data, May 1997, Tucson, Arizona, USA
12. A. Shoshani, L.M. Bernardo, H. Nordberg, D. Rotem, A. Sim, Multidimensional Indexing and Query Coordination for Tertiary Storage Management, 11th International Conference on Scientific and Statistical Database Management, Proceedings, Cleveland, Ohio, USA, July 1999
13. M. Wu, A.P. Buchmann, Encoded Bitmap Indexing for Data Warehouses, Proceedings of the Fourteenth International Conference on Data Engineering, February 1998, Orlando, Florida, USA
14. M. Wu, Query Optimization for Selections Using Bitmaps, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1999, Philadephia, Pennsylvania, USA
15. K. Wu, P.S. Yu, Range-Based Bitmap Indexing for High-Cardinality Attributes with Skew, Technical Report, IBM Watson Research Center, May 1996

A Fast Convergence Technique for Online Heat-balancing of Btree Indexed Database over Shared-nothing Parallel Systems

Hisham Feelifl[1] Masaru Kitsuregawa[1] Beng-Chin Ooi[2]

[1] Institute of Industrial Science, The University of Tokyo
3rd Dept., 7-22-1 Roppongi, Tokyo, 106-8558, Japan
{hisham, kitsure}@tkl.iis.u-tokyo.ac.jp
[2] Department of Computer Science, National University of Singapore
Kent Ridge, 119260, Singapore
ooibc@comp.nus.edu.sg

Abstract. In shared-nothing environments, data is typically declustered and indexed across the system processing elements (PEs) to achieve efficient processing. However access patterns are inherently dynamic and skewed, thus, data reorganization based on the data access history (heat) is essential and should be done online. While the data is being reorganized, indexes need to be modified too, therefore, reorganization should additionally deal with the index modification. Based on minimization of index modification, we propose a data reorganization technique over a shared-nothing parallel system. By finding the exact work that should be done, the technique can smoothly balance a given heat across the PEs as fast as possible, if it is required. By tuning its parameters, it can cover a wide range of balancing requirements. We evaluate its performance through simulation studies. Its effectiveness is clarified quantitatively.

1 Introduction

The explosive growth of data volume in various fields such as the Internet, Web, and, data warehouse increases the need for fast response. A shared-nothing parallel architecture is one of the typical examples to achieve such response [4, 13]. As demonstrated by the existing machines such as Bubba [2] and NEDO 100 Node PC Cluster [10], shared-nothing architectures can provide fast response at low cost, high extensibility and availability. However, shared-nothing architectures suffer from load balancing problems. Load (heat) balancing is difficult to achieve, compared with shared-disk architectures, because it relies on the effectiveness of database partitioning for the query workload [13]. To achieve efficient query (and transaction) execution, data is typically declustered across the PEs, and, indexed at each of the PEs. However, access patterns are inherently dynamic and skewed which can lead to performance degradation as some PEs become hotspots (frequently accessed) while many other PEs are cold (infrequently accessed). Thus heat-balancing is essential.

Heat balancing is particularly challenging for evolving workloads, where hot and cold data change over time. Data reorganization can only counteract such situations, and such reorganizations should be performed online without requiring the system to be quiescent [8]. As the data is moving from hotspot PEs to cold PEs, the

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 846–858, 2000.

corresponding indexes have to be modified too. Thus, data reorganization should also deal with the index modification [1].

In this paper, we propose a new technique to facilitate more efficient data reorganization. It is based on index-modification minimization, where the amount of data to be migrated (migration unit) corresponds to the entirety of one or more index branches at a source PE. In this case, it would be easy to prune the entirety of index branches from a source PE tree as well as attaching these branches into a destination PE tree using bulk-migration technique [5]. To distribute a given heat across the PEs, we introduce a new heat-balancing algorithm that is distinguished from the existing algorithms in several respects. It provides the exact solution to balance a system without any heuristic mechanisms. Thus it can distribute a given heat as evenly as possible across the PEs and as fast as possible, if it is required. Because of its exact solution, its convergence is guaranteed so that it can be employed in applications in which fast responses and fast balancing (adaptation) are relevant requirements. Furthermore, we support the technique by parameters that can be tuned to fit a wide range of balancing requirements in terms of heat distribution and balancing speed over a wide range of access pattern skew.

The next section details the related work. Section 3 describes the underlying index and the migration unit. Section 4 introduces a new heat-balancing algorithm. Sec. 5 deals with the experimental study and Sec. 6 concludes the paper.

2 Related Work

Recently, there has been much work in the area of online reorganization. [8] presents an efficient online method for the dynamic redistribution of data, however it does not cover index modification during reorganization. [7] outlines the issues involved in changing all references to a record when its primary identifier is changed due to a record move. The techniques of [7, 12] are proposed for centralized DBMS and require the use of locks, where using locks during reorganization can degrade performance significantly [1]. [1] presents two alternatives for performing the necessary index modifications, called one-at-a-time OAT page movement and BULK page movement. However, both techniques depend on the conventional Btree algorithms that can lead to considerable index-modification cost [5]. [11] suggests using the Fat-Btree structure to speedup migration issues so that index can be modified with minimum cost. However, the objective is to balance the number of pages across the PEs (space balancing) rather than heat balancing. Access pattern skew can lead to performance bottleneck even though there is a space balance [5, 13]. [5] assumes heat balancing but without consideration for the balancing speed (the speed of the system to adapt itself to an access pattern), where fast balancing (adaptation) is the main requirement for the most advanced applications, e.g. WWW servers. The balancing algorithm of both [5] and [11] is simply the disk-cooling algorithm [8] that can lead to a long convergence time and in some cases unstable situations as a result of its local view while balancing a system (see Sec. 4). In contrast, we avoid long convergence and unstable cases through an algorithm that utilizes the range-partitioning strategy and can find the exact solution in one step

without any heuristic mechanism so that the system is heat-balanced. By introducing balancing parameters into its solution, it can also cover a wide range of requirements.

3 The Global Index

We assume that data is initially range partitioned across all the system PEs so that the access method can associatively access data for strict match queries, range queries and can cluster data with similar values together. Using a B-tree based index enables more efficient processing of range queries compared to a hashed index because in the former only the nodes containing data in the specified range are accessed. One solution to associative access is to have a global index mechanism [6], conceptually, the global index is a two-level index with a major clustering on the PE's key range and a minor clustering on sub-ranges of the key range. This non-overlapping data partitioning gives also non-overlapping indexes [9], so that the first–level index can be implemented as a partitioning vector with entries number equals the PEs number. To ensure that there is no central PE, the first-level index is further replicated in all the PEs [5]. While the first-level index directs the search to the PE wherein the data is stored, the second-level index is basically a collection of Btrees, one at each PE; each Btree independently indexes the data at its PE [5, 9, 11].

We assume the tree height at each of the PEs is the same, so that the amount of data to be migrated corresponds to the entirety of one or more branches of the Btree at a source PE. The attachment of branches at a destination tree and detachment these branches at a source tree are essentially pointer updates so that index is modified with minimum cost. This branch-migration technique does not change the tree structure, but it changes the record distribution at a source and a destination PE. It causes an update in the root node of the Btrees at these PEs, which in turn requires the first-level index copies to be updated. This is can be done in a lazy manner by piggybacking update messages onto messages used for other purposes [5].

Since it is common to cache a part of the index in memory to accelerate access and cost per bit is declined, then it is possible to use large memory while data is migrated from a PE to other PE. Using of large memory for access method accelerates access, for example, [9] assumes that all the nodes of a Btree are placed in memory. Thus, if a root at a PE will overflow due to insertion of some branches (as a result of migration) then instead of having a physical fat structure the branches (and the corresponding data pages) that lead to fatness can be temporarily stored in memory. This saves considerable cost of I/O operations if the amount of migrated data is large, which can also lead to speed up migration issues among the PEs. To demonstrate this point, assume we have 4 PEs with the following key ranges: PE0 is assigned to hold 1-25, PE1 26-50, PE2 51-75, and PE3 76-100. If there is a migration decision that migrate some branches of key range say 15-25 from PE0 to PE1 and if the root node at PE1 is full, then it is possible to store these branches (and the corresponding data pages) into the memory of PE1. Then if it is required to migrate some branches from PE1 to PE2 of key range say 40-50, thus there will be rooms to accommodate the branches of key range 15-25 into the root of PE1 without need for being physical fat. While the branches of key range 15-25 are being stored in the memory of PE1, there is a

possibility to migrate these branches (or some of them) from PE1 to PE0, if it is required. Since there is no disk access, the corresponding migration cost is dominated by the communication cost between PE1 and PE0 and thus it speeds up migration issues between PE1 and PE0. Such buffering effect at PE1 can be generalized across the PEs so that it can provide fast access, and, migration at each of the PEs. Therefore using the PE memory if the fat condition is occurred at its root can reduce the cost of having a physical fat structure at a PE. The fatness property of the underlying index can be viewed as a temporary status that could be occurred at some PEs while data is being reorganized. Without loosing the generality we assume that the underlying index is basically Btree that can support bulk-migration without essence of being physically fat. However, to evaluate the performance of the proposed technique without any benefits due to buffering effect at each of the PEs, in our simulation we select the physical Fat-Btree structure [11] as the underlying index. So that the structure with its physical fatness represents the worst case of our reorganization.

4 Online Heat Balancing

Online heat balancing is done in four basic steps: monitoring PE workload, exchanging information between PEs, calculating new distribution and making the work migrating decision, and the actual data migration. In this section, we first clarify our consideration to the workload. Then we present a new algorithm to calculate a new heat distribution from the current one

The workload is reflected by a metric, called *heat* [3]. We define the heat of a range R = {Rmin .. Rmax} as the access frequency of R during a certain period of time. A range R as a logical quantity can be determined by any physical object in the system such as a data page, an index branch, and an index (sub)tree. The cost of maintaining heat statistics on R is dependent on its physical object. The highest cost can occur, if we maintain statistics for each of the data pages. This roughly requires maintaining statistics for every possible point in a given range. The minimal cost could be achieved if we maintain statistics for each tree (PE) in the system, and it requires information proportional to the number of PEs. Although it is simple, but it gives inaccurate estimation in the workload. There are mid-cost approaches, e.g., maintaining heat statistics for each index branch or for each sub-tree at a root node. These approaches give a compromise solution in terms of cost and accuracy. In our simulation, we use one such mid-cost approach in which heat statistics information is maintained for each sub-tree of a root node at a PE. To minimize the required information, uniform heat distribution is assumed in the deeper levels. In principle, we assume the workload estimation is a "design parameter" that depends on the applications and their requirements.

4.1 The Full-window Algorithm: Algorithm Basic

Since data is range partitioned across the PEs, we can only move data from one PE to its neighboring PEs which hold the preceding or succeeding ranges. This migration rule has two exceptions, the first deals with the *rightmost* PE, RMPE, which can only

migrate data with its left neighbor, while the second deals with the *leftmost* PE, LMPE, which can only migrate data with its right neighbor. Two main observations formulate the proposed algorithm; the first one is related to using the disk-cooling algorithm (DCA) [8] with the assumed range partitioning strategy, while the second is related to the migration exceptions at the RMPE and the LMPE.

Observation (1): Instability of the DCA.

To balance a given heat across the PEs, the existing techniques, such as [5, 11], use the DCA as the balancing algorithm. The DCA was introduced as an efficient general-purpose algorithm for balancing disk arrays. In the DCA, the hotspot disk is first selected as the migration source, and, the coldest disk is selected as the migration destination. By making this decision, it generates a new hotspot disk, if there is, at which the process can be repeated until all disks are heat-balanced. If we apply the DCA to the PEs with the assumed range-partition strategy, then, the procedure for selecting the coldest PE will be shorten as checking the right and the left neighbors of the hotspot PE. The colder neighbor will be selected as the migration destination.

The algorithm is simple but it has some unsatisfactory cases that can be occurred during balancing. For example, assume a system of 4 PEs with a heat distribution of (PE0: 200,PE1: 50, PE2: 115, PE3: 35) and a threshold heat of 110 (10 % above the average heat). The first scan gives PE0 as the hotspot, and heat of 90 is migrated to PE1 at which the heat will become 140. The second scan gives PE1 as the hotspot and its colder neighbor is PE0. Heat of 30 is migrated from PE1 to PE0. Thus some heat is returned back to PE0 by which the balancing process will be entered into endless loop and most of the work is consumed in useless migrations between PE0 and PE1 without distributing the given heat to other cold PEs. Such instability case occurs because the heat of PE2 is higher than the threshold heat which in turn blocks heat migration to other cold PEs, e.g. PE3. With the assumed range-partitioning strategy the DCA does not have a mechanism that can stop such useless migrations or a special mechanism that can deal with heat distributions that contain some blocking cases as in the given example. Such mechanism is important in shared nothing environments, where migrations of large data without any beneficial effect dramatically degrade their performance. Therefore, the DCA can not provide a fast balancing with full guarantee in its convergence and in general its local view to a system can lead to long convergence which slowdowns a system to adapt itself to a given access pattern. The observation gives the motivation to develop a new algorithm so that fast balancing can be achieved without doubt in its convergence and its convergence can further be tuned to fit a range of requirements.

Observation (2): The Rightmost and the Leftmost PEs.

Since the data are initially range partitioned across all the PEs, the RMPE must be heat-balanced by one of the following two cases:

1. If the heat at the RMPE is larger than the average heat, then it is required to migrate its excess heat to its left neighbor PE (LNPE).
2. If the heat at the RMPE is less than the average heat, then it is required to achieve its missing heat from its LNPE. .

In the first case, we refer to migration as a *direct migration*, since we can directly initiate heat migration from the RMPE to its LNPE, while in the second case, it completely depends on the heat at its LNPE as follows:

2.1 If its LNPE does not have such missing heat then it is required to achieve heat from other neighbors to the LNPE, which in turn suggests a recursive approach. We refer to this migration as *stacked migration*, because we can not initiate migration unless its LNPE has such heat. We push this information into a stack called the *migration stack* by storing its components: *source*, *destination*, and, *missing heat*.

2.2 If its LNPE has this missing heat; then migration can be directly initiated from the LNPE to the RMPE, and therefore we have a *direct migration* case.

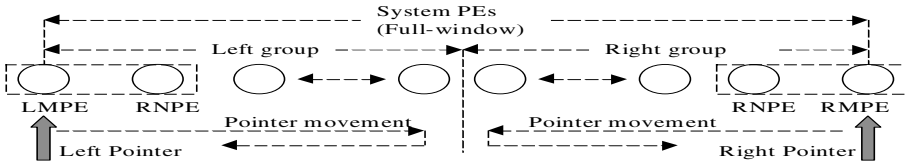
The above cases (from 1 to 2) find exactly the decision to balance the RMPE, and, they can be used also to find exactly the decision to balance the LMPE.

Based on this observation, assume we have a system of N PEs and we divide the PEs into two groups: left group and right group, see Fig. 1. We formulate the idea by first considering the system's RMPE (LMPE) with its LNPE (RNPE) and recording the proper decision to balance it, and, if it is necessary push the generated decision into the *migration stack*. Then, by dropping virtually these PEs from the system, there will be new RMPE and LMPE at which we can find the proper decisions to balance them with their neighbors as before, and therefore the process can be repeated until the system virtually consists of two PEs. At this point, it is easy to know exactly the migration decision that should be taken between the left and the right groups. Using the *migration stack*, we pop decisions by sequentially traversing the right group from left to right direction, and, the left group from right to left direction. Note that an empty stack indicates the definite algorithm termination. During traversing the PEs we store a generated decision into a structure called the *migration table* so that we can additionally know from the table what is the decision sequence to balance a system. Obtaining such sequence will help global balancing schemes in their scheduling to migration jobs while balancing a system.

Fig. 1 gives the algorithm notation, mechanism and high-level description. It shows two pointers called *right pointer* and *left pointer*, one for each group. The *right pointer* is initially pointed to the RMPE and it traverses its group from the right to the left direction. While the "left pointer" is initially pointed to the LMPE and it traverses its group from the left to the right. During these traverses, the current excess/missing heat at each pointer (PE) is recorded as well.

Case example

Assume a system of 8 PEs with the following heat distribution PE0: 200, PE1: 50, PE2: 115, PE3: 25, PE4: 260, PE5 20, PE6: 50 PE7 30. Assume further it is required to balance the system so that the heat at each of the PEs = 100 (average heat). The algorithm starts with its left pointer is pointed to PE0 while its right pointer is pointed to PE7. At PE0 a direct decision is generated as (source=PE0, destinnation=PE1, required heat =100) and the excess heat at PE1 becomes 50. At PE7 a decision of (PE6, PE7, 70) is pushed in the migration stack because PE6 does not have the missing heat of PE7 (70). The excess heat at PE6 becomes –120. By advancing both pointers, the new LMPE is PE1 while the new RMPE is PE6. At PE1 a direct decision




```
HeatType GetDecision(PE, Neighbor,ExcessHeat)
 // Balance the given PE with its neighbor. Identify
 // the generated decision by source, destination,
 // excess heat & type of decision (direct or stacked)
 if(Decision.Type=="Direct") Store(Decision);
 else PushDecision(MigrationStack,Decision);
 Calculate the new excess heat and return it.

void StackedDecisions(MigrationStack, MigrationTable)
  while(! EmptyStack(MigrationStack))
    Decision = PopDecision(MigrationStack);
    Store(Desicion); // in the migration table

Algorithm Full_Window (PeMin,PeMax)
// PEmin and PEmax covers the system PEs..Full window
   RP=PeMax;// Right Pointer pointed to the RMPE
   LP=PeMin;// Left Pointer pointed to the LMPE
   RightExcess=LeftExcess=0 //reset excess heats
   MigrationStack=CreateStack();Done=0;
   while(!Done)
    LeftExcess=GetDecision(LP,LP+1,LeftExcess);
    if(RP<=LP)Done=1;
    else{RightExcess=GetDecision(RP,RP-1,RightExcess);
         LP++;RP--;}
   StackedDecisions(MigrationStack,MigrationTable);
```

Fig. 1. The full-window algorithm: its notation, mechanism, and high level description.

Table 1. The generated migration table for the given example

Decisions Sequence	Source	Destination	Required Heat	Comment
1	PE0	PE1	100	Direct
2	PE2	PE3	50	Direct
3	PE2	PE3	65	Direct
4	PE4	PE3	10	Direct
5	PE4	PE5	200	Direct
6	PE5	PE6	120	Stacked
7	PE6	PE7	70	Stacked

of (PE1, PE2, 50) is generated and the excess heat at PE2 becomes 65. At PE6 a stacked decision is generated as (PE5, PE6, 120) and the excess heat at PE5 becomes –200. By advancing both pointers, a direct decision of (PE2, PE3, 65) is generated and the excess heat at PE3 becomes -10. A direct decision of (PE4, PE5, 200) is generated and the excess heat at PE4 becomes 10. The next step gives a direct

decision of (PE4, PE3, 10). Since each pointer completes its traverse to its group, then it uses the *migration stack* to extract the pushed decisions so far. Sequential pop operation completes the sequence to balance the system as given in Table 1. These pop operations are equivalent to traversing the PEs in the opposite direction to that of the first traverse, see Fig. 1. If this migration table is issued to the system, then in one step it distributes the given heat as evenly as possible across the PEs.

The full-window algorithm is basically distinguished from the DCA in two respects; its utilization to the assumed range-partitioning strategy, and, its global view to a system. Both give the exact solution to balance a system without any possibility of unstable cases while balancing. Consequently it gives the chance to do the required work as fast as possible (or as required) with full guarantee in its convergence. The above algorithm can be also extended in many ways, for example, instead of considering the whole PEs (full-window), it possible to derive other derived algorithms that based on some PEs (partial-window) which supports local balancing schemes rather than the global ones. We will consider such partial-window algorithms with their features and advantages in a future work.

4.2 The Migration-workload Parameters: Speed and Distribution

If a *migration table* generated by the above algorithm is issued to the given system then in one step it balances the PEs as evenly as possible. This one-step reorganization does the whole work in a short time that may be accepted by some requirements and rejected by the others, depending on their acceptance to its effect. Although the one-step reorganization gives the capability for the fastest adaptation to access patterns, but it can lead to slow responses (during reorganization) at some PEs at which there are large amount of data movement and high arrival rates of users' queries. During reorganization, users of such PEs are considered as the victims of the fastest reorganization. The number of victims increases as the skew of access patterns increases. This can be considered as the main disadvantage of the one–step reorganziation. Many requirements usually give some threshold in the required heat distribution and they also allow balancing in incremental ways rather than that of the one step. Thus, we view the requirement space as a 2-dimentional space of two parameters; one represents the speed requirement and the other represents the heat-distribution requirement. The current objective is to introduce such parameters in the balancing process so that migration decisions are correlated to a requirement.
Assume a migration entry of the *migration table* can be represented by:

```
typedef    struct {PeType   Source;
  PeType   Destination;
  HeatType RequiredHeat;
  void*    Others;} MigrationTableEntryType;
```

Since a system migration workload is proportionally related to the summation of the component "*RequiredHeat*" across the *migration table*, thus, controlling this component by some parameters (requirements) gives a direct correlation between requirements and migration decisions.

First we consider the speed parameter, α, we introduce this parameter into the component "*RequiredHeat*" by the following normalization (transformation):

"*RequiredHeat*"= α * "*RequiredHeat*", 0≤ α ≤1.0. So that for α=0, it implies dropping (or postponing) the encountered entry, while for α=1.0 it implies a full acceptance/speed for such entry. For 0<α<1.0 it implies we divide the current entry (job) into M small sub-jobs, where M=1/ α. If these M sub-jobs are issued one by one in periodic ways or whenever it is possible, then we achieve balancing in incremental ways, which certainly is reflected on a system speed to adapt itself to an access pattern. Although incremental balancing can lead to slow adaptation but it partitions the whole work into small jobs so that we can avoid the disadvantage of the one-step reorganization, especially under highly skew environments. Thus in general, any migration job can be postponed or incrementally issued or completely issued to a system depending upon its assigned α. For the sake of simplicity we select to unify α across the *migration table* so that one value of α gives one effect on the balancing speed (and the system response). Since the steady state of the M-step reorganization should be equal to that of the one-step as a result of its integration effect, then we can satisfy a heat distribution requirement (steady state) over a range of α.

Second we consider the heat-distribution parameter, ζ, we assume requirements on heat distribution are given as; balance a system so that heat at each of the PEs does not exceed some threshold value, (1+ ζ%) average heat. Note that the *migration table* has been constructed so far with the assumption of ζ=0. Because a ζ requirement gives the maximum allowable heat at hotspot PEs, we process the *migration table* by focusing on hotspot PEs. By picking up all the PEs that have heat higher than the threshold heat and modify (adjust) the component "*RequiredHeat*" so that the resultant heat at each of these PEs does not exceed the threshold heat. Then simulating the migration effect on the modified entries will generate new hotspot PEs by which the process can be repeated until a ζ requirement is satisfied across the *migration table*. This heat modification procedure can be sequentially done by; picking up the current hotspot PE and extracting a sub-table from the current table so that the sub-table contains all entries in which this hotspot PE is a source or a destination. We modify the "*RequiredHeat*" components in this sub-table so that the resultant heat of the current hotspot PE does not exceed the threshold heat. Simulating the migration effect in the current sub-table generates a new hotspot PE at which the process can be repeated until all entries in the migration table are modified according to the given ζ. The modification of the *"RequiredHeat"* component across the migration table implies also dropping some entries at which their sources are balanced in term of the given ζ requirement. The general structure of such procedure is;

```
FitZetaRequirement(MigrationTable,ζ)
  while((HotSpot=PickUpHotSpotPE(ζ))!=empty) do
     ST=ExtractSubTable(MigrationTable,HotSpot);
     while(E(ST)!=empty) do ModifyHeat(E.RequiredHeat);
     SimulateEffect(ST);
```

The introduced parameters α and ζ give the capability to cover a wide range of balancing requirements in terms of heat distribution and balancing speed. The requirements space includes the most restrictive ones, e.g. α=1, and, ζ=0. These basic parameters and the basic algorithm fulfill our objective of this paper. In the next section we report our simulation results.

5 Simulation Results

In this section, we describe our experiments to study the performance of online data reorganization using the full-window algorithm. We evaluate the system performance where the metric used is the impact on the response time of queries and the system migration workload. Table 2 shows the major parameters and their used values. We first create an initial Fat-Btree with the tuple key values generated using a uniform distribution (space-balanced). Then we generate range queries using Zipf-distribution skew defined by the skew factor (τ) of the Zipf-distribution. Thus, there are more range queries are issued at one PE than the other PEs, depending on the skew factor τ. A query range is selected to be equal that of an index branch so that the workload is based on the assumed migration unit (an index branch and its corresponding data pages). The heat skew initiates the migration of branches between the PEs, depending on the given ζ requirement. We model each of the PEs as a resource and the queries as entities. We assume heat balancing is done in centralized scheme and it is initiated every 100*N queries, where N is the PEs number.

Table 2. The major parameters and their values

Parameter	Default Value	Variation
System Parameters: Number of PEs in the cluster Network bandwidth Time to read or write a page	 16 120 Mbits/s 8 ms	 32, 64.
Database Parameters Number of records Index node size Data page	 2.1 millions (2MB) 4KB, key=4 Bytes. 4KB	
Query Parameters Zipf distribution of decay factor (τ) Hot spot location Mean arrival rate Mean service rate	 0.3 at PE0 20 500 ms	 0.1 $\rightarrow$ 0.9 PE0 $\rightarrow$ PE15 10 for skew variation experiment
Requirement Parameters. Speed parameter (α) Heat distribution parameter (ζ)	 1/4 10 %	 0, 1/64, 1/32, 1/16,1/8, 1. 0, 5, 15, 20, 30%.

It has been observed that balancing using the DCA leads to unstable balancing cases especially under skew of $\tau > 0.1$ which in turn limits our consideration to access pattern skew and balancing speed as well. Thus we exclude the DCA results from our discussion relying on the fact of the full-window algorithm always gives the exact solution without any unstable cases. In the first set of experiments, we study the effect of the full-window parameters (α, ζ) on the system performance. Figure 2 shows the full-window capability to fit a wide range of requirements including the most restrictive one (α=1.0, ζ=0.0). Figure 2.a traces the hotspot's response time, which indicates also the system capability to adapt itself to access patterns under various requirements on the balancing (adaptation) speed. Note that α=0 represents the hotspot response without heat balancing (space-balanced), while α=1 represents that

response with one-step reorganization. As shown this response can be controlled to a desired level by tuning the parameter α. We express the migration workload at a PE as the total time during which the encountered PE has been involved in migration issues as source or destination. Figure 2.b shows the migration workload at each of the PEs. The bell-like curves are obtained as a result of distributing heat by migrating index branches (and the corresponding data pages) from the hot PEs (e.g. PE0, PE1) to other cold PEs (e.g. PE10, PE11) through some PEs (e.g. PE3, PE4), in a ripple mechanism. As shown this ripple mechanism is mainly dependent on the ζ requirement, where ζ=0 represents distributing the given heat as evenly as possible (the highest migration workload). As requirements relax this restrictive heat distribution, the ripple-migration effect can be reduced accordingly.

To demonstrate the disadvantages of the one-step reorganization (α=1.0), we first trace the average response time at each of the PEs during reorganization and we record the PE responses that almost dominate the system response during reorganization. As shown in Figure 3 (left), PEs other than the hotspot like PE2 and PE4 have slow responses because at these PEs there are high migration workloads and high arrival rates of users' queries. Such effect under the considered skew (τ=0.3) can not be avoided with the given requirement of the fastest balancing. Users of PE2, PE4, and PE6 during reorganization are the victims of the fastest balancing. However, if requirements allow incremental balancing under high skew environments, then by lowering α, e.g., α=0.25, such effect can be avoided as shown in Figure 3.(left). It shows the system response is dominated by the main hotspot PE, as a result of partitioning the whole work into 4 steps. The experiment indicates also the advantages of the incremental balancing under high skewed environments.

With the assumed range partitioning strategy, it has been observed that the system migration workload (and response) is dependent on hot-spot locations in the system, where the RMPE and the LMPE do not have much freedom in their migration direction like the other PEs. Figure 4 affirms such dependency on hotspot locations, where we express the system migration-workload as the summation of the migration workload at each of the PEs. It shows also that the ratio of the maximum migration workload to its minimum is about 2.5. This high ratio indicates the main problem of the given range-partition configuration which is mainly selected to simplify the implementation of the first-level index, and consequently, the search operations. However it gives us the motivation to consider another configuration in a future work so that migration decisions are mainly correlated to access patterns skew rather than to their favorite locations in a system. It shows also that we select the hotspot at PE0 (the worst case of the migration workload) to evaluate the proposed technique.

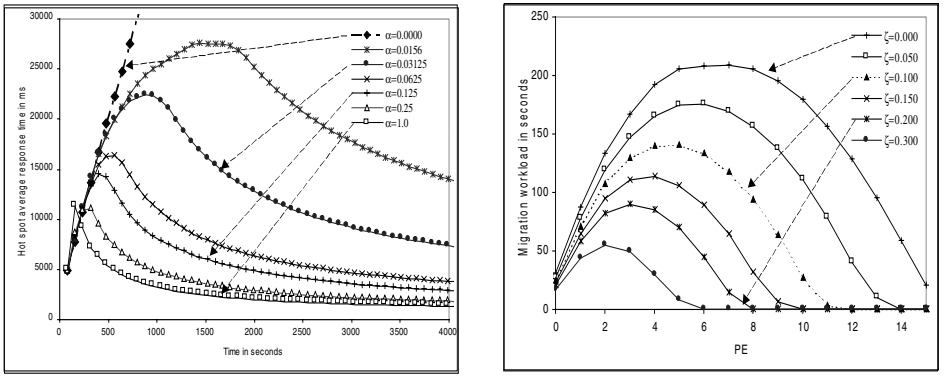
To demonstrate the scalability of the proposed strategy, we study the system migration-workload under different environments of skewed access patterns, and, number of PEs. Figure 5 shows that as the access pattern skew increases the system's migration workload increases in a nearly linear relationship. It demonstrates also the superiority of the full-window algorithm in dealing with access patterns over a wide range of skew. We repeated the experiment for different numbers of PEs for clusters of 32 and 64 PEs. It shows that as the number of PEs increases, the migration workload increases which in turn emphasis the need for heat balancing.

6 Conclusion

In this paper, we have developed a heat-balancing technique for Btree indexed database over shared-nothing parallel systems. It demonstrates that finding the exact solution for heat balancing avoids unstable cases while balancing, and, long convergence time. This gain gives a system the capability to adapt itself to access patterns as fast as required. Through its parameters, the technique can cover a wide range of balancing requirements in terms of heat distribution and balancing speed over a wide range of access pattern skew. It can be used for data placement with the goals of optimal system performance and it is useful for dealing with both advanced DBMS such as office document management or WWW servers, and relational database systems. In these application data declustering can be exploited by an appropriate mapping of documents/records into index keys. Apart from complexity, the simulation results are a first step toward gaining quantitative insight into the performance of our technique. Developing techniques that automate data placement in shared-nothing systems is a crucially important problem. We believe that the proposed technique is a promising approach toward solving this problem.

References

1. Achyutuni, K. J., Omiecinski, E., Navathe, S. B.: Two techniques for On-line Index Modification in Shared Nothing Parallel Databases. Procs ACM SIGMOD (1996)
2. Boral, H., et al: Prototyping Bubba, a Highly Parallel Databse System, IEEE Trans. On Knowledge and Data Eng., Vol. 2, No. 1, March (1990)
3. Copeland, G., Alexander, W., Boughter, E., Keller, T.: Data Placement in Bubba. Proc. of ACM SIGMOD Conference, pages 99-108, (1988)
4. DeWitt, D.J. and Gray, J.: The Future of High Performance Database Systems. Communication of ACM, 35(6), 85-98, (1992)
5. Lee, M. L.,Kitsuregawa, M., Ooi, B.C., Tan, K, Mondal, A.: Towards Self-Tuning Data Placement in Parallel Database Systems, Proc.. ACM SIGMOD pages 225-236 (2000).
6. Ozsu M., Valduriez., P.: Principles of Distributed Database Systems, Prentice Hall, (1991)
7. Salzberg, B., A. Dimock. Principles of transaction-based on-line reorganization. Procs. of the 18th Inter. Conf. on VLDB, pages 511-520, (1992)
8. Scheuermann, P., Weikum, G., Zabback, P., Adaptive Load Balancing in Disk Arrays. Proceedings of the 4th Inter. Conf. FODO, (1993)
9. Seeger B. and Larson P. Multi-Disk B-trees. ACM SIGMOD Conf.1991, 436-445
10. Tamura, T., Oguchi, M., Kitsuregawa, M.: Parallel Database Processing on a 100 Node PC Cluster: Case for Decision Support Query Processing and Data Mining. Proc. Of SC97: High Performance Networking and Computing, (1997)
11. Yokota, H., Kanemasa, Y., Miyaazaki, J.. Fat-Btree: An Update-Conscious Directory Structure. Procs. of IEEE the 15th IEEE Conf. on Data Engineering, pp. 448-457, (1999)
12. Zou C., Salzberg, B.: .On-Line Reorganization of Sparsely-Populated B+ Trees. Procs. ACM, pages 115-124, (1996)
13. Valduriez, P.,: Parallel Database Systems: Open Problems and New Issues, Distributed and Parallel Databases 1, No. 2, 137-165, Kluwer Academic Publishers, Boston, MA (1993).



(a) (b)

Fig. 2. The effect of the full-window parameters (α, ζ) on **(a)** the average response time of the hot spot PE (α effect) **(b)** the PE migration workload (ζ effect).

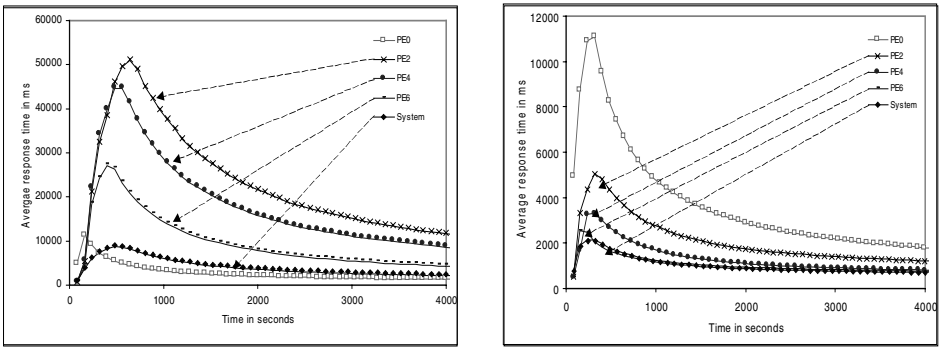


Fig. 3. The effect of the one-step (left) and M-steps (right with α=0.25) reorganizations on the PEs that dominate the system response.

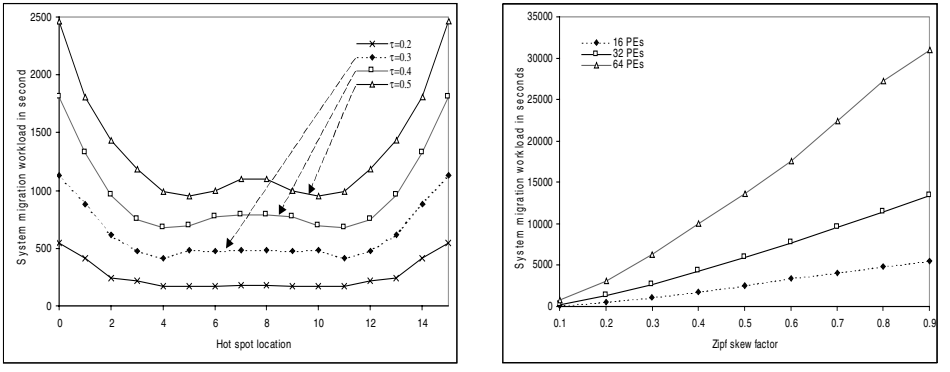


Fig. 4. The effect of the hotspot location on the system migration workload.

Fig. 5. The scalability of the proposed technique.

Indexing Semistructured Data Using PATRICIA Tree

Li-Cheng Wu, Jorng-Tzong Horng, Baw-Jhiune Liu, Chin-Yea Wang, and Gwo-Dong Chen

Dept. of CSIE, National Central University, Taiwan
Richard@db.csie.ncu.edu.tw
http://db.csie.ncu.edu.tw/~richard

Abstract. Information on the Web like HTML documents with images, video, and sound is a collection of heterogeneous data. HTML documents are semistructured in nature. Semistructured data are used to describe those structures which are less rigid or regular than those data found in standard database systems. This study presents a novel means of using Patricia Tree [14] to index semistructured data. This index is used by transferring the query into a regular expression and querying the regular expression over the Patricia Tree. The highlight of this approach is supporting query on content and structure simultaneously, while also supporting fast query time on long path and regular expressions.

1 Introduction

The amount of data through networks has dramatically increased in recent years owing to the popularity of the World-Wide-Web. The data are usually linked with Hyper Text Markup Language (HTML), which is either poorly structured or totally unstructured. Information on the Web are often accessed through documents written according to the HTML specification. HTML documents are semistructured in natural. According to the definition of [1], the data that is irregular or that exhibit type and structural heterogenity is semistructured data. Semistructured data are used to describe those data with an implicit structure, and which are less rigid or regular than those data found in standard database systems [1]. Information on the Web like Fig. 1 including texts, images and even audio and video is treated as semistructured data. As SGML/HTML becomes the standard of document exchange, the schema information in the document can be retrieved using a proper makeup. Fig. 1 presents a sample of document data describing a movie company. Using proper markup document type definition, the document in Fig. 1 can be represented with markup language in Fig. 2. Since each movie of the movie company may exhibit some differences, the structure of each part may not be the same despite having a proper markup. For example, the movie in the document has a different state from finishing status. Consequently, when creating a single table for the document in the movie name of Fig. 1, it will be problematic to determine which name should be recorded for movies with one old name and one new name. A further problem arises in creating a table to

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 859–868, 2000.

Fig. 1. Sample document of a company profile

Fig. 2. Markup language of the SGML in Fig. 1

record information such as "Chief Eric has a picture printed on Times." It is not easy to save a record for this information because Chief Eric may have a picture printed on Time or may have never had another picture on any magazine. Such a unfix structure makes it difficult to determine whether if a new table should be created since it may always contain only one row of data, the attributes may always be insufficient, or the table content may be full of NULL values. In general semistructured data like this is hard to be stored and queried in relational or object-oriented database systems easily and efficiently.

Stanford University proposed a semistructured data model called Object Exchange Model (OEM) [15], and developed a system Lore [2] as a prototype database system to store semistructured data. Data in OEM is thought of as a labeled directed graph. Fig. 3 shows the OEM graph for the semistructured data shown in Fig. 1. The vertices in the graph are objects. Every node in the graph has a unique object identification number (OID) and is stored in the object base. The object content can be retrieved once the OID number is specified. The reader may refer to [15] regarding the details of the model. In this paper, we don't focus on the model and borrow the model to describe semistructured data.

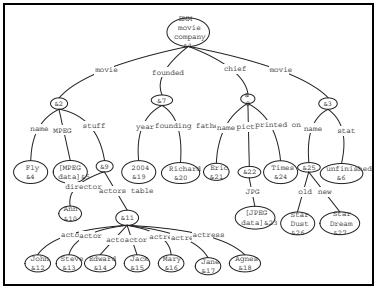


Fig. 3. Semistructured data schema of document in Fig. 1

In a DBMS, indexing is an important technique as it allows fast access to data. Indexes have shown themselves to be a useful technique despite the added storage, the cost of index maintenance, and the added complexity in the query engine. The motivation involving the use of indexes for semistructured data is similar in a database systems. Indexes in relational or object-oriented database systems are created over a set of attributes where the types of the attributes are defined in advance. But semistructured data has an arbitrary data type, so it is difficult to determine an attribute of a collection to index.

The indexing method in Lore [16] comprises of four different indexes: a value-index indexing atomic values, a text-index for full text searches, a link-index for locating links, and a path-index for locating edges. The Lore architecture and query execution engine are described in [2]. Lore's cost-based query optimizer was introduced in [12] with a focus on how the optimizer finds efficient query plans. Lore DataGuides [9], which are dynamic database schema, serve both as a tool for the end-user and as a simple path index. In [5], a notion of "state extent" is introduced for path expression evaluation that resembles Lore's DataGuide path indexes [11]. Both DataGuide [9] and Tindex [5] extract the common paths on the schema which is not very efficient on very irregular schema. Patricia Tree [14] is a text indexing and supports approximate matching [4] and searches for regular expressions [3]. Traditional Patricia index [6] focus on searching documents and did not use the indexes on semistructured data although some improvement has been made to make Patricia Tree suitable for the database [17].

This study presents a novel means of using Patricia Tree to index semistructured data. The semistructured data paths are transferred into strings and the Patricia Tree technique is used to build an index. This index is used by transferring the query into a regular expression and querying the regular expression over the Patricia Tree. The highlight of this approach is supporting query on content and structure simultaneously, while also supporting fast query time on long path expression and regular expression.

This paper is organized as follows. Section 2 describes how the semistructured data is indexed by Patricia Tree. Section 3 describes how query processing can use the index to speed up query processing time. Section 4 draws a conclusion.

2 Index Structure

Conventional information retrieval uses Patricia Tree to index pure un-structured text data. The OEM [15] data model is designed to model semistructured data. To apply the Patricia Tree algorithm, the tree must be converted into strings. How can Patricia Tree be reversed to index semistructured data? By reversing the path strings, the prefix string will be duplicated. A novel method of indexing semistructured data is developed by incorporating reversed path strings into Patricia Tree. With reversed path strings, the content of the string will be located in the front of the string, making the content index selectable. Meanwhile, the path string of the leaf node will contain that of the parent node, making the index size smaller and allowing the query to still be achieved in the same

algorithm. An OEM data model models a semistructured database as shown in Fig. 3. Each node has a unique object identification for object retrieval. To simplify the presentation of index creation, Fig. 4 shows a partial schema from Fig. 1 using this representation. Each node in Fig. 4 is represented by the label of incoming edge and OID separated by the "/" symbol. Every node to the tree

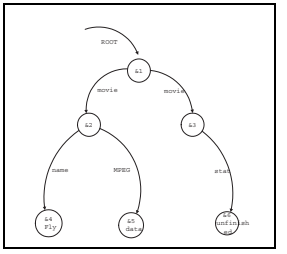


Fig. 4. A partial OEM schema of Fig. 3 shown

root represents an identical path. For example, Fig. 4 contains six nodes. The path strings are given as follows:

path string 1: root/&1
path string 2: movie/&2.root/&1
path string 3: movie/&3.root/&1
path string 4: Fly:name/&4.movie/&2.root/&1
path string 5: data:MPEG/&5.movie/&2.root/&1
path string 6: unfinished:stat/&6.movie/&3.root/&1

Notably, each path string covers that of the parent node. In the above example, path string 2 and 3 contain path string 1 as sub-string. The next step converts these path strings into sistrings[7]. Sistring is short for "semi-infinite string", made by cutting strings into many sub-strings. Because after the "." character the sistrings are duplicated with sistrings of the parent nodes, sistrings after the first "." character are cut down. Taking path string 4 for example, eleven sistrings exist from the first character to the eleventh character "." . The sistrings of path string 4 are given below.

sistring1: Fly:name/&4.movie/&2.root/&1
sistring2: ly:name/&4.movie/&2.root/&1
sistring3: y:name/&4.movie/&2.root/&1
...
sistring11: 4.movie/&2.root/&1

The sistrings after twelve will be in the sistring of path string 2 and path string 1 should not be inserted twice. Since the OID differs for every node, the path string will differ even when the path name is the same. For example, two paths share the same name "root.movie" in the schema, but their sistrings ,"movie/&2.root&1" and "movie/&3.root&1", will be different. This arrangement ensures the same path, such as "root.movie" will be inserted into the index twice with different OID, and will successfully be retrieved by the query. The details of insertion process of Patricia Tree can be found in [7]. The index is created with all sistring inserted into the Patricia Tree, and thus the steps are

repeated for all sistrings. Via the compression method in [17] the tree can be compressed into three arrays. Using this compression method, the Patricia Tree can be stored on the disk structure with a space efficient method.

3 Query Processing Using Indexes

In this Section, we introduce how a query is processed using the patirica Tree constructed in Section 2. The index method proposed herein supports keyword query, path query and path/content query simultaneously.

3.1 Query Processing with Index Support

The query processor uses the following two procedures when executing a query with index support.

1. Transfer a query to a regular expression
2. Transfer regular expression to DFA and obtain the query result

The query processor receives three types of queries, i.e., path query, keyword query, and path/content query.

1. Path query: The query clause contains a path expression requiring retrieval.
   Example: INPUT (movie.name) OUTPUT: ({2} {4}) ({3} {25})
   INPUT(chief.name) OUTPUT : ({8}{21})
2. Keyword query: The query clause contains a keyword that matches in either path or content.
   Example : INPUT(Fly) OUTPUT : ({4})
   INPUT(Agnes) OUTPUT : ({18})
3. Path/content query: The query clause contains both path expressions and node content.
   Example: INPUT(movie.name=Fly) OUTPUT: ({4}{2})
   INPUT(Founding Father=Richard) OUTPUT : ({20}{7})

The index we propose supports these three types of query to rapidly locate the result. To use the Patricia Tree to search the results a query needs, a query clause should be transformed into an intermediate form of a query for the Patricia Tree search. For example, the query clause "root.movie.name=Fly" will be transferred to "Fly:name&?.movie&?.root" after adding the needed information to the query clause. The query clause is translated to NFA. The NFA is then changed to DFA. Finally, DFA is simulated on the Patricia Tree search. This makes results that match all requirements possible. The steps in [7] and [18] are used to transfer a regular expression to DFA and get the query result.

3.2 Sample Queries

Several sample queries are used in this subsection to demonstrate how three types of queries are processed. The queries use the semistructured schema shown in Fig. 3

Path query The query clause, "root.movie", is taken as an example. First, the path string is converted to "movie&?.root". Because this query clause only contains the path expression, the converted string does not have a node/path separator. The path is converted into binary DFA and simulated on the Patricia Tree. If the Patricia Tree is correctly created with the sistrings, then both "movie&2.root" "movie&3.root" will be final states of the binary DFA, and OID lists ({2} {1}) ({3} {1}) will be returned.

Keyword query The example query clause is "movie". The string "movie"is directly converted into NFA. For this query, the Patricia Tree acts as a simple full-text index engine except that it not only indexes the content but also the path names. After simulate two sub nodes and the OID {2} {3} will be returned. Matching keyword is faster than regular expression since the tree traversal is involved only.

Path/content query We take query clause "root.movie.name=Fly", as an example. First, the path string is converted into "Fly:name&?.movie&?.root". Because the regular expression being converted is slow on "?" wildcards, using some "OR" operator and avoiding "?" will accelerate the process. To accelerate the query processing, if the OID is all numbers, then "0-9" can be used, meaning this character may be 0 ,1 ,2 ,. . . , 9. So path string can be replaced by "Fly:name&(0-9).movie&(0-9).root", and OID ({4} {2} {1}) will be returned.

3.3 Performance Evaluation

In this subsection, we evaluate the performance of query processing with index support. The experimental platform is an PC with Intel Pentium II© 450 with 128 Mb RAM. The queries are randomly selected from document sets, executed ten times each and the resulting times are averaged. The source data is classified into character system and the fan-out of the data, which is a statistic of how the data schema are dispersed. We use two data sets on your experiments. The DBCS dataset which is html documents from the web site of Linux Fab[1] , a web site full of linux technical hypertext written in Double Byte character set. SBCS dataset is the TREC 5 record which is more regular since DTD is fixed. An average of the data statistics is used to measure data dispersion. On simple hierarchical document, the formula in [13] can be simplified as follows. For every path p appearing in the database:

- Total number of instances of path p is denoted $|p|$.
- Total number of distinct nodes reachable via p is denoted $|p|d$.
- Total number of l-labeled sub-object reachable from p is denoted $|pl|$

[1] http://linuxfab.cx

Thus, the fan-out [13] is P $= |p| * (|p|d/|pl|)$. Total fan-out is averaged as parameter of path p and the average is calculated. For every l reachable from p, average fan-out is defined as

$$(\sum_{l} |p|(\frac{|p|d}{|pl|})/|l|$$

Since the fan-out of a path p has been obtained, the average total fan-out of all paths p can also be obtained. The total fan-out is summed and the average is calculated. as an parameter of path p. Average fan-out of data with

$$\sum_{p} ((\sum_{l} |p|(\frac{|p|d}{|pl|})/|l|)/n$$

Clearly, if the data base has a distinct name path, the fan-out will be greater. Fig. 5 presents the execution time for simple queries with a path-expression like X.Y.Z="L" on the SBCS data set, using the Patricia Tree, the path index with an inverted list, and no index, respectively. According to Fig. 5, the Patricia Tree is only slightly affected when data fan-out increases grows. The path index with inverted list index will encounter problems when data fan-out increases. The data indicates that as the fan-out exceeds 1.2, the Patricia Tree becomes superior as a data index. This result confirms that Patricia Tree performs better when data fan-out is larger. The experiment is also tested on the DBCS data set. It obtains the similar results. The path performance of the path-index and Patricia Tree is tested. The test query only contains the path information. Fig. 6 presents the execution time for simple queries with a path-expression X.Y.Z on SBCS data set, using the Patricia Tree, path index in [11] and no index, respectively. According to Fig. 6, the Patricia Tree is little affected while data fan-out is increasing. The path index will encounter problem while data fan-out is increasing. The data indicates that as the fan-out enlarge, the Patricia Tree becomes preferable as a data index. This result in turn indicates that the performance of the Patricia Tree comes mainly from fast response time in comparing path index in [11]. The result of the DBCS character set obtains a similar result and is not shown. The Patricia Tree can serve as a text-index which locates the

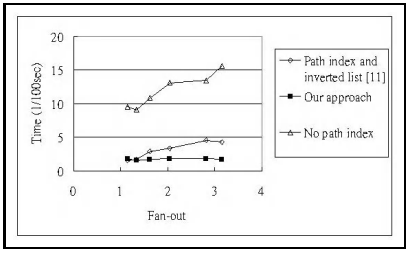


Fig. 5. Execution time vs. fan-out on the SBCS data set

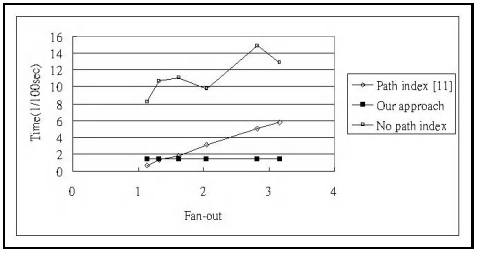


Fig. 6. Path query execution time vs. fan-out on the SBCS data set

keyword location in the database. Fig. 7 presents the execution time for a simple queries of randomly chosen keywords on the SBCS data set, using the Patricia Tree and inverted list. The test result in Fig. 7 is different from that in Fig. 5 and hard to make a conclusion on which one is better on different fan-out data. Since the inverted list uses hashing function, an unstable performance is possible. Fig. 7 also explains that the Patricia Tree on the keyword remains slightly affected when the data fan-out is increasing. There may be more stable result on larger scale experiment on inverted list. Thus, the performance of the inverted list in different fan-out data on this figure cannot be concluded. Fig. 8presents

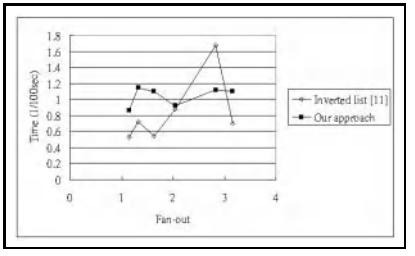


Fig. 7. Keyword query execution time vs. fan-out on the SBCS data set

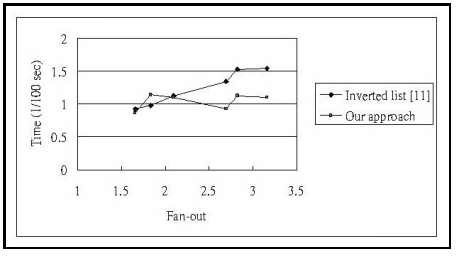


Fig. 8. Keyword query execution time vs. fan-out on the DBCS data set

the execution times for simple queries keywords on an DBCS data set, using the Patricia Tree and inverted list in [11]. The test result in Fig. 8differs from that in Fig. 7. Since the DBCS character set does not resemble the SBCS character set in being easy to divide into word alignments, the inverted table is difficult to build. Although the inverted list will use a hashing function, the DBCS character set inserted too many records into the inverted table and the performance deteriorates further while the fan-out increases and the characters to be inserted grow. Fig. 9 presents the operator executing time of different length path. The

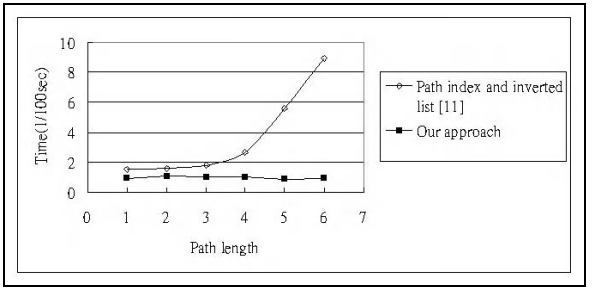


Fig. 9. The execution time of different path lengths

experiment shows that the Patricia Tree was not affected by the long query path

length. Theoretically, the query time for the Patricia Tree should increase with long query length, but observing the execution time reveals it is dominated by generating the regular expression and the simulation time is not the bulk of the operation execution time. The path index will require joining operations on the long path, and consequently the execution time on the long path on the path-index is not very good. As the semistructured database grows, the structure of data may become looser and path length will increasingly important to the query. Patricia Tree will help make the long path to retrieval more efficient.

4 Conclusion

This study presents a novel means of using the Patricia Tree to index semistructured data and support querying content and structure simultaneously. Semistructured databases are a developing area in database systems and new index methods are being studied. Herein, a path index to index semistructured data and achieve fast query response time is developed. The proposed method can be used by Stanford's query optimizer and as a fully functional query operator. The performance is evaluated and it performed acceptably well when data fan-out is large. Restated an index is being created for a database with very large schema and large fan out, then using the index method proposed to index the database is recommended. Some application databases have these features, such as program code libraries, medical information databases, and document databases.

Table 1 completely lists the comparison of different index approaches.

Table 1: The comparison of our approach and path index with inverted list

	Advantages	Drawbacks
Our Approach	Not affected data dispersion Wildcard keyword support Indexes both structure and content Regular expression on path expression Suitable on all path lengths	Require more memory Slower for well-structured data
Path index with inverted list	Fast on short paths Index size is smaller Faster for well-structured data	Require four indexes Slower on unstructured data Slower for longer paths

Reference links and backward links remain a problem with this approach and work is in underway by using pre-parsing method. The Patricia Tree supports longest repeat and proximity searching which may provide more information for optimization as a further research issue.

Acknowledgments The authors would like to thank the National Science Council of ROC for financially supporting this research under Contract No. NSC 89-2213-E-008-006.

References

1. S. Abiteboul. Querying Semistructured data. Proceedings of the International Conference on Database Theory, pages 1-18, Delphi, Greece, January (1997).

2. S. Abiteboul, D. Quass, J. McHugh,J. Widom, and J. Wiener. The Lorel query language for semistructured data. International Journal on Digital Libraries, 1(1):68-88, April (1997).
3. R. Baeza-Yates and G. Gonnet. Fast Text Searching for Regular Expressions or Automaton Simulation over Tries. Journal of ACM, 43(6) November (1996), 915-936.
4. R.Baeza-Yates and G.Gonnet. A Faster Algorithm for Approximate String Matching. Combinatorial Pattern Matching (CPM'96), Irvine, CA, LNCS 1075, Jun (1996), 1-23.
5. M. Fernandez and D. Suciu. Optimizing regular path expressions using graph schemas. In Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, February (1998).
6. G.H Gonnet. Examples of PAT applied to the Oxford G. Navarro.A language for queries on structure and contents of textual databases. Master's thesis, Dept. of Computer Science, Univ. of Chile, April (1995).
7. G.H Gonnet, R.A. Baeza-Yates,and T.Snider New Indices for Text:Pat Trees and Pat Arrays, in Information Retrieval Data structures and Algorithms. In Information Retrieval: Data Structures and Algorithms. Prentice Hall, Englewood Cliffs, Newjersey (1992).
8. G.H Gonnet, R.A. Baeza-Yates,and T.Snider. Examples of PAT applied to the Oxford English Dictionary. Technical Report OED-87-02, UW Centre for the New OED and Text Research,Univ. of Waterloo, (1987).
9. R. Goldman and J. Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In Proceedings of the 23th International Conference on Very Large Data Bases,436-445, Athens, Greece, August (1997).
10. D. Knuth The art of Computer Programming. Sorting and Searching. Addison-Wesley,Reading,Mass.,(1973).
11. J. McHugh, J.Widom, S. Abiteboul, Q.Luo, and A. Rajaraman, Indexing Semistructured Data. Technical Report, Stanford University Database Group, (1998). http://www-db.stanford.edu/pub/papers/semiindexing98.ps.
12. J. McHugh and J. Widom. Query optimization in semistructured data. Technical report, Stanford University Database Group, (1997). http://www-db.stanford.edu/pub/papers/qo.ps.
13. J. McHugh and J. Widom. Optimizing Branching Path Expressions Technical report, Stanford University Database Group, (1999). http://www-db.stanford.edu/pub/papers/mp.ps.
14. D. Morrison. PATRICIA-Practical algorithm to retrieve information coded in alphanumeric. JACM, 15:514-534, (1968).
15. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In Proceedings of the 11th International Conference on Data Engineering, pages 251-260, Taipei, Taiwan (1995).
16. D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information. In Proceedings of the 4th International Conference on Deductive and Object-Oriented Databases (DOOD), Singapore, December (1995).
17. M. Shishibori, M. Okuno, K. Ando and J. Aoe An Efficient Compression Method for Patricia Tries . (1997) IEEE International Conference on Systems, Man, and Cybernetics.
18. M. Sipser. Introduction to the Theory of Computation. PWS Publishing Company. (1997)

MegaStore: Advanced Internet-based E-Commerce Service for Music Industry

Ammar Benabdelkader, Hamideh Afsarmanesh, and L. O. Hertzberger

University of Amsterdam, Faculty of Science Research Institute Computer Science
Kruislaan 403 1098 SJ Amsterdam, The Netherlands
{ammar, hamideh, bob}@wins.uva.nl

Abstract. To support necessary requirements and flexibility to the buyers of different goods; advanced and efficient internet-based Electronic Commerce services must be designed and developed. In addition to the traditional user requirements, the developed system must properly address efficiency issues, among which the data catalogues, information classification, short response time for on-line requests, high system performance, and high data transfer rates must be considered. The MegaStore[1] system described in this paper, aims at the design and set-up of the necessary database structure and platform architecture for advanced e-commerce applications. The proposed system architecture best suits the e-commerce application, by separating the public information from the private information and supporting the large data sets that need to be securely kept at predefined Internet sites. The design of the innovative architecture and technology for the distributed MegaStore application although applied to music CD industry, is general enough to be applicable to other complex application environments, especially to e-commerce applications.

1 Introduction

One major technical problem hampering the realization of suitable implementation for electronic shopping is the lack of possibilities to integrate a wide variety of data in a single coherent environment, which bases on a comprehensive system architecture and advanced database technologies [1], [8]. In the context of web-based systems, several applications have been investigated and worked out during the last few years. These applications cover different domains of interest and address several application requirements. As such, these applications address different domains ranging from simple search engines that allow users to find information of their interest using a user friendly interfaces [11], [12] to advanced systems that manipulate multimedia information taking into account the emerging Internet technologies [7], [13], [14]. Nowadays, more e-commerce applications embracing electronic shopping via virtual stores are also emerging [16], [17]. However, Web-related systems are still involving the development of a large number of tools for data manipulation [9], [10] and require a lot of efforts for their internal maintenance and operation. The work described in

[1] The Virtual MegaStore project has been supported by the Dutch HPCN foundation whose partners are the University of Amsterdam, the Frame Holding BV, and the International Music consortium.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 869–878, 2000.

this paper overcomes some of these problems and addresses the need to provide the user of electronic shopping with an environment through which he can experience as sufficiently close to real life shopping experience.

From the usage point of view, the Internet-based CD shopping system (called the *Virtual MegaStore)* consists of a *front-end* system with two main components. The first component being the *Internet-Shop,* that can be accessed by all Internet users and the second component constituting a so-called *Shop-in-a-Shop* interface. The *Shop-in-a-Shop* interface can be installed inside an existing physical music store and it offers the store keeper the unique and strong ability to immediately and at run-time respond to the requests of customers visiting the music shop, through downloading the raw music data and producing CDs tailored to the customers requests.

The MegaStore *front-end* system bases on a *back-end* component that includes a distributed object-oriented database and a high performance server architecture. The database supports geographically distributed multi-media information and the designed extensible server architecture assures the required high data transfer rate and the short response time for on-line requests.

The structure of this paper is organized as follow. In section 2, the music industry application is studied and analyzed, thus, the necessary requirements are determined. Section 3 describes and designs complex music library information, and accomplishes the MegaStore base schema structure using the UML notation. Section 4 focuses on the general design of the server architecture for the MegaStore system, and mainly outlines the necessary requirements for the *Internet-Shop* interface, the *Shop-in-a-Shop* interface, the data volume estimation, the parallel/distributed server extension, and the data storage mechanisms. In section 5, brief descriptions of different music formats supported by the MegaStore system are presented. Section 6 outlines the current implementation status of the system. And section 7 concludes the paper and foresees possible extensions to the system.

2 Problem Analysis and Required High Level Architecture

The analysis of the music data to be stored and transferred between music shops and the users, plays an important role in defining the MegaStore server architecture. The MegaStore network must be designed and build in such a way that it provides high bandwidth for huge amount of data transfer in a very short response time while taking into consideration the information visibility rights and security of access. A specific characteristic of the MegaStore application is that the real music data is kept by certain music label centers, and neither can it be centralized in one common database, nor can it be freely or randomly replicated at different sites.

To properly support the requirements of the MegaStore environment, the designed system architecture involves the following components: [2]

1. The *back-end* system, including the database engine and the predefined networking connection between the MegaStore system components.
2. The *front-end* system, including (1) the *Internet–Shop* interface, where a user from home (or work place) can search for music, listen/watch to the audio/video clips, and order CDs, and (2) the *Shop-in-a-Shop* interface, where the music storekeeper

can fetch on-line the real music data from its original source in order to burn at run-time the requested music CDs.

Under the specifications of the MegaStore system enumerated above, we have identified the need to design and build a very dedicated networking infrastructure for this application, where the following aspects are carefully studied:

- The music data is geographically distributed over the network
- Information about music is classified into two main categories: the general information stored at the Directory Services that can be accessed by any user connected via the *Internet-Shop* Interface and the raw music data that can only be accessed by the music storekeepers at music centers or burning towers.
- Depending on the user profile and authorization, only a part of the information can be accessed, and users need not to know about the data distribution.
- The real music data is securely transferred through a dedicated Network among music centers.
- The system must benefit from intelligent caching mechanisms, which is now being further investigated at the University of Amsterdam, in order to improve the performance of the system [4], [5].
- High bandwidth connection is necessary to handle raw music data that needs to be passed between the real music storage centers and the burning towers.
- Low latency network connection for the *Internet-Shop* interface is necessary to support the huge number of users expected to connect to the system.

3 Database Design

The database design for MegaStore is achieved in collaboration with the experts in the music industry domain. For the schema modeling, mnemonic names are chosen, taken from the music context, and thus objects are named for what they represent. This choice helps for instance the storekeepers to easily understand the elements of the database schema and use that in formulating their requests.

The dynamism and flexibility of the *Virtual MegaStore* system mostly depends on its database design and how open it is in supporting several application domains with different structures and different size. Different pieces of information about the *MegaStore* application domain are defined and stored as a set of inter-linked objects of different kinds, grouped by their domain of interests, e.g. artists, songs, CDs, consumers, stores, burning towers, etc.

The schema represented in Fig. 1 shows the static view of the MegaStore system in terms of classes and relationships among them. The name of a class is derived from the problem domain and must be as unambiguous as possible. The attributes define the characteristics of the class and capture the information that describes and identifies the class; every attribute has a type, which specifies what kind of data it represents. The relationship association between classes is drawn as a solid line and has a name and a multiplicity range [18].

The part of the database schema design for the MegaStore system that is presented in Fig. 1 describes the detailed structure of its Directory Service. For instance, a customer can order some albums, where each album consists of a set of Songs, and it is possible that one or more artists sing every Song. A song may also

have a link to its music composer, music performer and/or some instruments. Such a specification may help in satisfying users through many points of views. For instance, a user may be more interested in his/her search, in the music performer, the music composer, or the used instruments, rather than being interested in a search based on the artist name or the song title.

The *Album* entity represents the class for CDs, tapes, and other means of music titles collection. Mainly, an album has some characteristics, consists of a set of songs, and one or several artists sing the songs in the album. The class *Album* links to other classes such as *Artist* and *Song,* through the specified relationships "*Album Artists*" and "*Album Songs*".

The class *Song* represents the main entity in the MegaStore system. Within this class, the complete information about each song is defined. The richness of MegaStore system strongly depends on the availability of such information in the database. Since the *Internet-Shop* is a dynamic Web interface, for which Web pages are created on the fly depending on the user request, the system automatically checks the database and provides the user with the most complete information that it finds in the database.

The class *Song* has three links to the classes *Album*, *Artist*, and *Instrument* via the defined relationships "*Song Of Album*", "*Sung By*", and "*Uses*".
The class *Artist* is the entity that holds all the information about each artist. Under the normal consideration some attributes such as the artist name, artist photo, and a short biography, are enough for the artist description.

The class *Customer* keeps the necessary information about the customers. Each time a user makes an order, the system automatically checks the user's identity based on the information available in the database, and decides whether to directly access the information about him/her from the database, if it exists, or requests it from the user, if it is not.

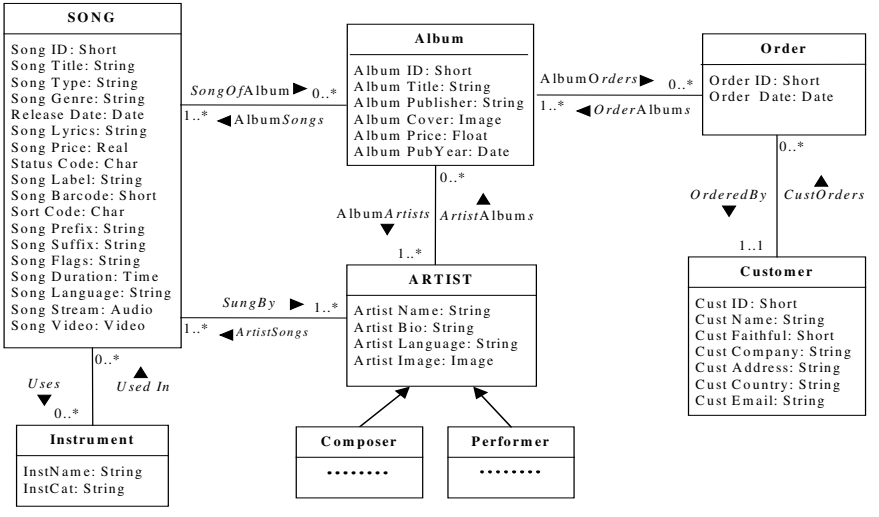


Fig. 1. Schema definition for the MegaStore System

4 The MegaStore System

This section focuses on the design of the server architecture and its extension to support the *Virtual MegaStore* system. The choice of the server architecture extensions are due to the need for supporting the information management and the data transfer requirements.

As depicted in Fig. 2, there will be three kinds of connections that need to be established between the components of the MegaStore system, namely:

1. High bandwidth connection, which is required for the case of transferring a considerable amount of data. This is usually the case, for transferring raw music data between geographically distributed music centers where the studies show that a minimum of 1 Gigabyte per second Internet connection is required [2].
2. Medium bandwidth with a medium latency connection, which is required for transfer of medium size data between the MegaStore system components. This is usually the case, for updating the directory services when new music albums are produced (e.g. the connection from the *Shop-in-a-Shop* interface to the directory services).
3. Low latency connection, which is required to support the huge number of high end-users who access the MegaStore system and require the transfer of small amount of data (e.g. the connection between the Internet-shop interface and the Directory Services). The analysis, of this application domain, shows that a huge number of users must be supported [2], the data to be transferred between the Internet user and MegaStore server however, will be in the range of medium to small size.

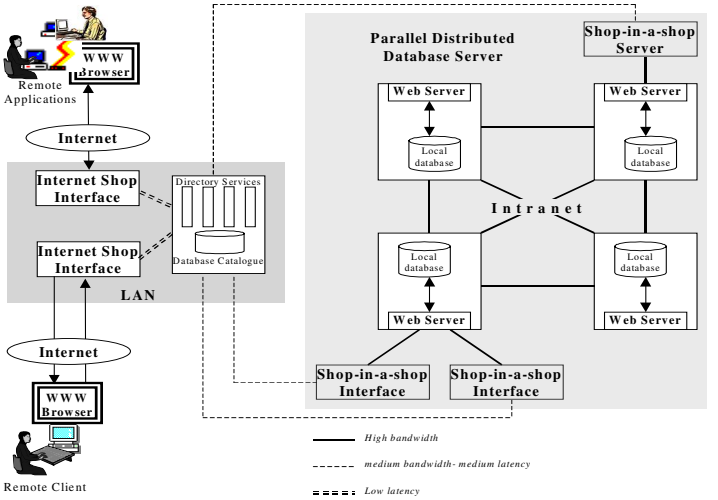


Fig. 2. MegaStore Server Architecture description

4.1 The Internet-Shop Interface

The *Internet-Shop* attempts to give the user the same feeling as when he or she is visiting a real music store. The music store, as we know it today, is a place where items can be touched, where intuition plays a bigger role than knowledge, and where music albums are usually bought in an unplanned fashion. This is how most people behave during shopping and this is also what makes the shopping itself fun. The real shop is not limited to a search engine, which finds products in a faultless but also emotionless manner. Even for the more expensive products in general, the final decision is often not objective, but is based on the emotions caused by the color, form, brand, and price. Therefore, the user interface of the *Internet Shop* needs to invoke the same kind of emotion as in a real shop.

This means that in an ideal case, a consumer from home wishes to request tracks/titles that he/she wants to be included in one CD of his/her own compilation, and if he decides to buy this tailored CD, he wishes to pay, in a secure way, by means of electronic payment. If the payment is accepted, the system must automatically allocate this customer's order to the closest music shop, in relation to the location of the customer. At that shop the titles are eventually burned and delivered.

4.2 The Shop-in-a-Shop Interface

The *Shop-in-a-Shop* interface adds a new dimension to music shopping. Normal music shops only have a limited stock. If a customer is looking for a specific piece of music that is not in the stock, he or she cannot be served. To avoid this problem, the *Shop-in-a-Shop* system places a *customer terminal* either in the music shops or other shops, e.g. photo developing shops, or even supermarkets, that supports the browsing and selection of music that is stored in a database server. If the customer has decided to purchase a piece of music, a CD is produced by downloading the raw music data from the database server. Also, additional information like booklet, the CD label, and the CD cover is downloaded and produced. Naturally, the music store must have a very high bandwidth Internet connection to be able to retrieve the large raw CD data in a few minutes time. The advanced internet technology although perhaps lucking behind the support for vastly increasing number of customers for e-commerce, due to the costs and required efficiency, dedicated fast connections are expected to be available in the near future to properly support the music industry application. Also, the production equipment like CD burners and high quality printers are expected to be sufficiently fast in the near future.

4.3 Data Volume Estimation

This section gives some estimation about the data volume that needs to be handled within the *Internet-Shop* and the *Shop-in-a-Shop* interfaces.

For the *Interne-Shop* interface, the average disk space requested per CD will be around 8 Mg (a CD album contains about 15 titles: 550 KB * 15 = 8.25 Mg). Thus, a prototype system of 5000 albums requires about 40 Gigabytes of disk space [2].

For the *Shop-in-a-Shop* interface we should add to the total disk space needed for the *Internet-Shop* an average of 650 Mg per CD representing the complete raw data, which is around 3500 Gigabytes for a system dealing with 5000 albums.

The MegaStore system is designed in a flexible way that supports different implementation strategies. For instance, if the system needs to be extended to support more albums and titles, one strategy that preserves the system performance is to store the audio/video clips together with the raw music data at the distributed/parallel database server and not at the Directory Services. This approach not only extends the system in term of supporting a considerable amount of music data, but it also preserves the performance of the system including short response time and data security, by keeping the directory services as efficient as possible. The Directory Services plays a major role in defining the general MegaStore information and specifying where the related raw data for each song or album is located within the distributed system.

4.4 Server Architecture Extension

This section addresses the server architecture extension to support both the information management and data transfer requirements for the MegaStore system. The main requirements to take into consideration includes:

- Design and implement the extensions needed for the existing parallel server system [6], in order to support all identified *Virtual MegaStore* database functionality. These extensions support:
  - The functionality needed by the HTTP daemons (Web server) front-end, in terms of support for the Web user interface, including the streaming of audio and video data.
  - Easy database administration.
- Develop and implement a mechanism that supports the entry of music and associating data into the database system [3].

Distributed Parallel Server Extension. To provide the MegaStore web server with efficient access to the raw music data, a parallel/distributed database framework is designed and developed [6]. With this implementation, the nodes (music stores) of the distributed MegaStore server are inter-connected, making it possible for specific users to connect to any node in the distributed server and to request an object, without the need to know where that object actually resides.

Due to the music data specification and copyrights that do not allow data replication or redundancy, this data must be securely kept at the site where it belongs (data location is known before hand).

The distributed database supports the following required functionalities:

- Provides a way for managing huge amount of data
- Data is securely kept at geographically distributed music centers
- Data is stored only at the point(s) where it belongs
- Data is visible from any node (music center) within the cooperation community
- Data is efficiently transferred between the nodes in short response time

Data Storage and Manipulation. The MegaStore data manipulation concerns two components: the database catalogue at the Directory Services and the parallel-distributed database server at the back-end system. All the music information including the short clips of the converted streaming audio/video that do not require high security protection will be placed at and accessible through the Directory Services, to be made available to the Internet users. However, the real raw music data that serves for CDs burning is securely kept at different distributed music centers linked to each other via a secure network connection, so that only authorized users can access and manipulate the raw music data.

The music data loading for MegaStore is a two-fold process. On one hand it stores the raw music data at the secure distributed server, and on the other side it updates the Directory Services with the general information concerning newly acquired albums and titles. The storage of the music data is provided locally at each site (music centers) by the music producer framework that can be in some cases integrated with the *Shop-in-a-Shop* interface. At this level, in order to keep the system up-to-date and more consistent, not only the music data entry mechanisms are provided, but also the music data conversion and data formatting are considered [3].

5 Music Audio and Video Content

This section addresses some issues related to the music data conversion and briefly describes the variety of different music formats supported by the MegaStore system, as well as it provides information concerning the music encoders.

Music Audio and Video clips consist of previously captured digital audio or video files, which can also be recorded from many types of media device [21]. Currently, the MegaStore System supports most of the existing audio and video formats including Real Audio, MPEG, CD Tracks, Waveform, QuickTime, etc.

In addition, the MegaStore system is open to support other emerging standard formats, such as the Secure Digital Music Initiative (SDMI) [19]. However, most efforts investigated on music data conversion are basically focussed on the Real Audio [21] and the MPEG [20] formats, due to the various advantages of these two technologies over the others. The RealAudio has an advantage of producing both audio streaming and video clips, and the generated files are of smaller size. The MP3 however presents the advantage that it produces near CD-quality music and it is widely used over the world.

When audio files need to be processed or digitized, an encoder and an encoding algorithm must be selected. Most Encoders can encode using several different algorithms. Each encoding algorithm is optimized for a particular type of audio and connection bandwidth [22]. Such a dynamic selection of audio/video clips allows the system to provide the Internet-user with the best quality connection his/her system can handle, without the user having to explicitly choose from separate clips recorded for different speeds. In the MegaStore system, for the digitized music clips we used RealAudio 56K ISDN, Music – Mono and Stereo. This template best suits our needs since we expect Internet users via ISDN connection.

6 Current Implementation Status

For the implementation purpose of the MegaStore system, we intend to combine different technologies to support the application requirements as described in section 2. Based on the detailed study of the MegaStore application functionality needs, the appropriate approach to apply and the technologies to use are identified [3]:

- The designed database is being implemented on top of the Arches machines at the University of Amsterdam[2].
- The Matisse database system [15] is used as the object-oriented distributed database system (ODBMS). Among other features Matisse supports transaction management, concurrency control, historical versioning, indexing mechanisms, high speed for data access, multi-media streaming, and standard programming interfaces using C/C++, Java, ODBC, OQL, etc.
- Different Internet infrastructures are deployed depending on each functionality of the MegaStore system (a high bandwidth communication between the music stores for huge amount of data transfer and a low latency communication to the Directory Services for high end-users access).

Some experiments are performed using an NT front-end machine to run a DBA interface and an Internet-Shop server, that is in turn connected to the Matisse database running on the Arches machines, using the ODBC driver. The database administrator (DBA) interface, implemented in C++ and Windows NT environment, provides some administration facilities including the automatic loading of the music data and the creation of the necessary links between inter-related pieces of information. And The Internet-Shop server is implemented using a combination of the most recent and relevant software technologies including JAVA Script and Visual Basic Script for tips programming, Active Database Objects (ADO) for database connection, and HTML for text formatting. The implementation of the server is made possible, using the Active Server Pages programming environment that allows the combination of all these different software technologies in one single environment.

7 Conclusion and Future Work

This paper addresses the innovative design methodology of an open architecture for the MegaStore application. The paper first describes in details the application analysis and the database design, and then addresses the Internet-Shop and the Shop-in-a-Shop interfaces. Furthermore, it discusses the issues of music conversion mechanisms, and the distributed server extension for the MegaStore system. The application analysis and the database design were achieved in collaboration with the experts from the music industry. Thus, the database schema description and names chosen for schema components were taken from the music context, and object names in the MegaStore are mnemonic. To provide the MegaStore web server with the efficient access to the raw music data, a distributed/parallel database framework is adapted and extended to handle the huge amount of raw music data required for burning Compact Discs.

---

[2] The Arches system is currently composed of 20 nodes containing each a dual Pentium II with 512 MB ram and 9 GB disk, and it supports several network communications.

We believe that the MegaStore framework is more than a system for music titles and albums. The main idea behind the developed framework is to design a comprehensive system to support applications with two specific characteristics: to hold large data sets and to manage multimedia information. Thus, the MegaStore system can be considered as a general implementation approach, from which other application in biology and medicine that share the same characteristics can benefit.

8 References

1. Benabdelkader, A., Afsarmanesh, H., Hertzberger, L.O.: Database Support for Multimedia Information in Web based Applications. Int. Conf. on Computer Technologies MICCT'99, Tizi Ouzou Algeria, (1999) 169-184
2. Benabdelkader, A., Afsarmanesh, H., Hertzberger, L.O.: The Virtual MegaStore System Architecture: Analysis and Design. Tech. R. CS-99-04, Faculty of Science Research Institute Computer Science University of Amsterdam (1999).
3. Benabdelkader, A., Afsarmanesh, H., Hertzberger, L.O.: The Virtual MegaStore System Implementation. Tech. R. CS-99-05, University of Amsterdam (1999)
4. Belloum, A., Hertzberger, L.O.: Replacement Strategies in Web Caching. Int. Conf. ISIC/CIRA/ISAS'98, Gaithersburg Maryland USA (1998).
5. Belloum, A., Hertzberger, L.O.: Dealing with One-Timer Documents in Web Caching. Int. Conf. EUROMICRO'98, Vastega Sweden (1998)
6. A., Peddemors, A.J.H., Hertzberger, L. O.: A High Performance Distributed Database System for Enhanced Internet Services. 6th Int. Conf. HPCN, the Netherlands (1998)
7. Bouguettaya, A., Benatallah, B., Ouzzani, M., Hendra, L.: WebFINDIT - A System for Querying Web Databases", Int. J. IEEE Internet Computing (1999)
8. Bouguettaya, A., Benatallah, B., Hendra, L., Beard, J., Smith, K., Ouzzani, M.: World Wide Database - Integrating the Web, CORBA and Databases. Int. Conf. ACM - SIGMOD, ACM Press, Philadelphia USA (1999)
9. Beeri, C., Elber, G., Milo, T., Sagiv, Y., Shmueli, O., Tishby, N., Kogan, Y., Konopnicki, D., Mogilevski, P., Slonim, N.: WebSuite -- A tool suite for harnessing Web data. Int. Workshop on the Web and Databases WebDB'98, Valencia Spain (1998)
10. Falquet, G., Guyot, J., Nerima, L.: Language and tools to specify hypertext views on databases. Int. Workshop on the Web and Databases WebDB'98, Valencia Spain (1998)
11. Nortel, Kazman, R.: Web Query: searching and visualizing the Web through connectivity. 6th Int. World Wide Web Conf., Santa Clara CA (1997)
12. Mills, T., Moody, K., Rodden, K.: Providing World Wide Access to Historical Sources. 6th Int. World Wide Web Conf., Santa Clara CA (1997)
13. Abiteboul, S.: Views and XML. Int. Conf. of the ACM Symp. on Principles of Database Systems (1999) 1-9
14. Mukherjea, S., Hirata, K., Hara, Y.: Towards a multimedia World Wide Web Information Retrieval. 6th Int. World Wide Web Conf., Santa Clara CA (1997)
15. Matisse Database Manuals (ed.): Copyright © 1996 ADB S.A, 3rd Edition, France (1998)
16. CD Now, http://www.cdnow.com
17. Amazon, http://www.amazon.com
18. Eriksson, H.-E., Penker, M. (ed.): UML Toolkit. John Wiley & Sons Inc., New York Chichester Weinheim Brisbane Singapore Toronto (1998)
19. SDMI: Secure Digital Music Initiative, http://www.riaa.com/tech/tech_sd.htm
20. MP3 Encoders, http://www.musicglobalnetwork.com/encoders.html
21. Real Audio Player, http://www.real.com
22. Real Encoder, http://www.real.com/help/encoder/mac3.0/encoding.html

Supporting Public Browsing of an Art Gallery Collections Database

Sean Bechhofer, Nick Drummond, and Carole Goble

Department of Computer Science,
University of Manchester,
Oxford Road, Manchester M13 9PL, UK
seanb@cs.man.ac.uk,
WWW home page: http://img.cs.man.ac.uk

Abstract. Increased public awareness and usage of the Web suggests that a commensurate Web presence is required from providers of cultural heritage information such as Galleries and Museums. While galleries have traditionally supplied goal-driven search facilities to specialists, they must now provide browsing and query facilities to casual users, with less precise information seeking requirements. Hypertext systems provide an appropriate technology to support the networks of associations required in order to provide path-based browsing. Requirements are twofold: browsing support for the users; and authoring support in the creation of pathways. In our prototype, we combine techniques from Hypertext and Information Retrieval to provide access to artifacts drawn from the costume collection of the Manchester City Art Gallery. We provide similarity based browsing, using terms from the artifacts' metadata to calculate similarities. The approach is simple, yet effective as the results of a user evaluation show.

1 Introduction

In the past it has been difficult for members of the public to access information due to a lack of available resources, libraries or expert help. However, galleries and museums are under pressure to increase public access to collections, and technologies such as the WWW have made information less restricted by medium and more widely available. Increased public awareness and usage of the Web suggests that a commensurate Web presence is required from providers of cultural heritage information.

The Manchester City Art Gallery of Costume at Platt Hall, was the first institution in this country to be solely concerned with the history of clothes. The collections held by the Gallery are extensive, covering all aspects of dress, dress care and dressmaking, as well as aids to the appearance. There are associated collections of textiles, embroideries, lace and dolls. The total number of items in these collections is approximately 19,400, including 12,850 items of costume, 3,800 OPUA (objects of personal use and adornment), 2,400 textile items and 376 dolls or items of dolls' clothing. There are in addition approximately 1,600 unaccessioned items used for study and as handling collections.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 879–888, 2000.

Along with many other galleries and museums, the Gallery is investigating the use of IT as a way to improve public appeal and effectiveness as an information provider.

The Gallery's visitors fall into three main categories:

1. Goal-driven specialists who are seeking specific information and have a precise information need;
2. Casual but goal-driven seekers, for example students given a specific assignment, that have an imprecise information need;
3. Casual visitors who are not goal-directed and have an imprecise information need.

While galleries have traditionally supplied search facilities to specialists, they must now provide browsing and query facilities to casual users of category 3, whose information seeking requirements are imprecise and less goal-driven. The truly casual user will have no idea of the content of the collections, nor of how they might be categorised, classified or indexed. Not only is their information need vague, but they would not be able to articulate it with any degree of confidence of acquiring a result if they could [5]. Access to collections must be tailored towards identified audience needs.

A considerable body of research exists on information seeking [4, 14, 22], predicated on the analysis of the interaction of an expert mediator, e.g. a gallery specialist or librarian, with a searcher. A common scenario is that the visitor has some idea of the kind of artifact they would like to look at, and they describe this to the gallery front-desk advisor. These descriptions vary widely in form and clarity, including concrete requests: *the Sunflowers by Van Gogh*; simple "kind of" categories: *a wedding dress*; vague requests, using abstract categories: *something glamorous*; and complex, narrative descriptions where the customer "paints a picture" of what they would like: *19th century men's evening suit.*

Analytical search, as defined by Pejterson [19] is supported on the WWW by search engines, ideal for goal-driven precise retrievals, such as concrete requests or possibly narrative description. They are less ideal for supporting search by analogy or browsing, though they can be used to identify a sub-collection that can be browsed.

Our focus here is primarily on browsing [17] – an undirected navigation around the collection of objects, taking short steps from point to point. Browsing provides an overview of a space, requires a smaller cognitive load than an analytical search strategy and perhaps most importantly in this context, can aid in discovery and learning. In this case the overview provides an idea of the content of the collection. *Across-document browsing* allows the user to gain a sense of the form of the collection – a casual user in the gallery is unlikely to invest time and effort in a specific search.

In order to support such browsing, the crucial question that the system must address is "*where can I go from here?*" or "*what other things like this are there?*". To support navigation, documents should be dynamically linked together in a range of diverse ways; pathways, such as Walden Paths [10] should be formed that

act as trails through the network. Users have diverse perspectives [6]. It must be possible to locate information from a range of entry points, and associate information that is similar in a variety of ways. Information can be richly linked and *classified* into multiple clusters simultaneously, based on some shared notion of similarity. Hypertext systems would appear to be an appropriate technology to support the complex networks of associations required to support path-based browsing. Thus requirements are twofold: better **browsing** support for the users; and better **authoring** support for the dynamic creation of browsing pathways.

The paper describes an experiment into how such an information provider can present information in an effective, efficient and satisfactory manner, focusing in particular on the task of public access browsing. In our experimental prototype, we draw on and combine techniques from Hypertext and Information Retrieval to provide scalable access to artifacts drawn from the costume collection of the Manchester City Art Gallery. The specific browsing paradigm we follow is one of analogy based browsing, using terms in the artifacts' metadata to calculate similarities.

2 Hypertext

The word "hypertext" was first introduced by Nelson [18] although its origins can be traced back to the Memex of Bush [7]. A hypertext consists of a collection of linked nodes or documents. The user navigates from node to node, traversing or following the links. The gallery objects and their descriptions in the collection form a hypertext – browsing the objects in the collection is a form of link-following.

2.1 First Generation Hypertext

A first generation hypertext system consists of a collection of nodes or documents along with manually authored, static, untyped point-to-point links. Although such systems (for example the WWW) are extremely powerful and offer flexibility, they suffer from several problems. Maintenance is difficult, particularly if the number of nodes is high, or the information represented by the nodes changes frequently. If a new node is to be added, we may have to add many new links to pre-existing nodes. Similarly, deletion of nodes can lead to problems of "dangling links". This is a crucial issue as we expect the system to be extensible as further objects are added to the collection. If we are to use a hypertext model as a delivery mechanism for our gallery browsing, broken links are highly undesirable.

The link structure in a first generation system is prescriptive – the only links available are those provided by the system designer, resulting in systems which are less able to adapt to the user. The lack of any semantics or types on the links can also lead to difficulties in interpreting the structure or defining how the hypertext should behave. Although a hard-wired static predetermined links structure has its advantages (particularly if the intention is to educate the user

or provide a guided trail through the collection), if it is the only navigational facility offered, it constrains the reader [12]. The designer has to "second-guess" the requirements of the user and consider all eventualities. The issue of link types is particularly important as objects may have different kinds of associations between them.

The major issue is scale. The difficulty of creating large numbers of manually-authored links limits the size of the hypertext. A concrete example is found in typical WWW public access systems that do not reflect the diversity of user perceptions, and where there is limited scope for locating information from a range of viewpoints. Categories are poorly interconnected – frequently, beneath a couple of categorisation layers, the information structure breaks down to simple lists, offering limited scope for following up an interest. Where keyword searching exists, the onus is on the user to determine suitable terms. If the request gives no answer, the user must determine which keywords to try next.

2.2 Second Generation Hypertext

Second generation hypertexts have attempted to address these shortcomings through the use of more principled authoring. In particular, there has been a move to separate the links from the documents, and specify the structure and behaviour in terms of a well-defined conceptual schema, typing documents and links. Information about the hypertext is now represented explicitly from the information in it. Links between documents can be derived by querying the schema, allowing richer and more elaborate associations.

A considerable amount of research has been applied to the automatic construction of hypertext structure. However, the time to supervise link creation for large and growing collections is prohibitive [21]. Moreover, too many links can overwhelm the reader.

Such a view can be taken to its limit, with all access to objects being provided through query rather than linking. This is more the view of the Information Retrieval (IR) community, where the system maintains a descriptor for each document or object that characterises its content. Queries consist of a composition of descriptors and the system returns those documents matching the query. Navigation from document to document is conducted entirely through queries. Link creation using IR methods has been applied to hypertexts to suggest links to authors, to inform retrieval algorithms to retrieve hypertext nodes, and to infer links among documents (e.g. [15, 16]). SuperBook [20] used keyword queries instead of static hypertext links as a navigation mechanism, although this relied on a table of contents interface metaphor that we consider inappropriate. See [11] for a list of references.

2.3 Implementation Strategy

Our approach employs a second generation architecture, with links derived from the content of the object descriptions, forming a document space and a link space of the kind described in [1, 6]. We create links automatically based on

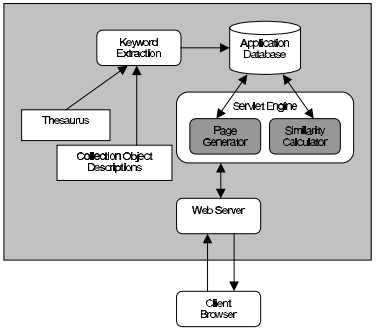


Fig. 1. System Architecture

the content of documents describing the artifacts in the costume collection and take the simple approach of extracting keywords (taken from a thesaurus) from those descriptions. The question "*what is there like this?*", is then answered through an examination of the extracted keywords and a comparison with other objects in the collection. Our experimental prototype is intended to test the hypothesis that such a straightforward approach will produce an adequate and usable browsing environment for casual exploration of the collections.

Manually authored links still play a part in providing browsing facilities. The system designers may wish to provide particular trails through the collection. Such a mechanism should be seen, however, as one of a number of complementary navigational devices, which can be employed and should be incorporated in the system.

3 System Architecture and Implementation

The system maintains a database of objects along with their descriptions and other catalogue data. In addition to the textual descriptions, we generate keyword annotations based on those descriptions. The annotations are then used to determine the "similarity" of objects in the collection.

Figure 1 shows the basic architecture of the system. The use of an HTML based front end will allow us to develop the application for WWW usage, providing remote access to the collection. In addition, with the growing public usage of the Web, familiarity with navigational mechanisms such as clicking on links or images to see further information is widespread. The database (stored as an XML document) contains basic information about each object in the collection as shown in Figure 2. Not all fields (such as donor or date) are present for each object.

Keywords are drawn from a thesaurus of terms related to costume. The thesaurus has a simple structure [2], including broader, narrower and related terms, and is based loosely on the ICOM Costume Classification [13]. A Keyword Analyser and Extractor tokenizes the descriptions applied to the objects,

Field	Purpose
id	Unique identifier
accession number	Gallery information
name	The object's name
date	Date information
donor	Where the item came from
picture	A link to an image of the object
links	Objects in the database considered to be closely related
description	A textual description of the object

Fig. 2. Database Fields

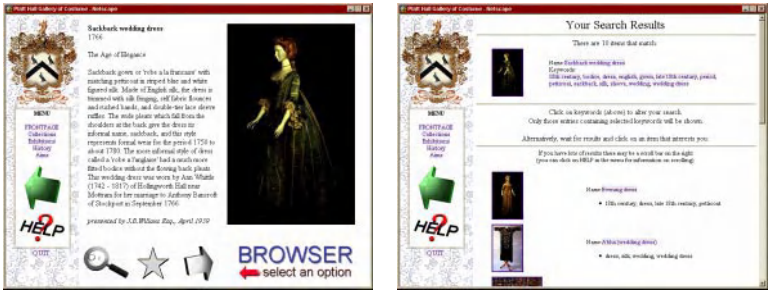


Fig. 3. Application Screenshots

applies stemming to extract and match terms against those in the thesaurus and annotates each entry with the terms that it finds.

Servlets generate HTML descriptions for particular objects, which are then delivered to the client browser. The results of a page generation are shown on the left of Figure 3. The page has a frame at the left providing access to the front pages and a tool bar along the bottom, which provides access to linked or related objects. For example, selecting the magnifying glass yields the page shown on the right of Figure 3. The system has found 10 objects matching the focus, and displays them in rank order (those with most matching keywords at the top). A list of the keywords applying to the object is shown, and users can use these to refine the list, in a sense combining the browsing and searching notion. However, this is not a straightforward search, but is a form of search by analogy, along the lines of Query-By-Example [23], where the focus object provides an initial query, which can then be refined.

4 User Evaluation

Evaluation of browsing is a difficult task. Standard quantitative measures such as recall and precision can be used to evaluate performance of search systems, but in our particular application, they are less appropriate. Instead, we conducted a qualitative evaluation, and assess the satisfaction that users felt when using the system through a questionnaire.

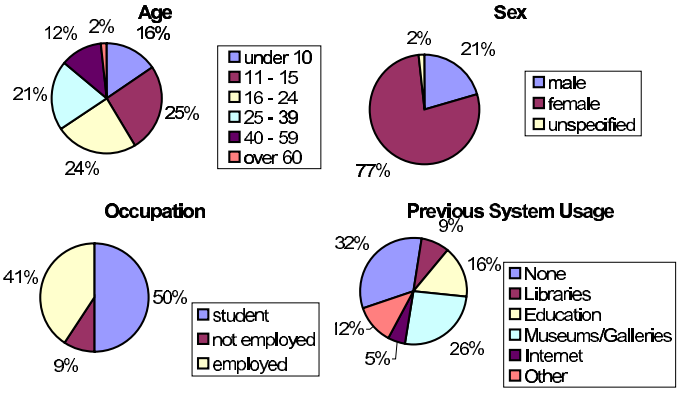


Fig. 4. User Demographics

4.1 Questionnaire

Users of the system were asked to fill out a questionnaire requesting their name, age, sex, occupation status (employed/unemployed/student), whether they had used any similar information systems in the past and if so, where. They were then asked to rate how the system compared to others they had used (if appropriate) and whether they found the system easy to use and useful. Responses to these latter questions were based on a seven-point scale. Additional information about the users was gained through interview and observation.

The trials were conducted at the Platt Hall Gallery over a three-week period. Platt Hall is located in South Manchester, close to the Universities and many local primary and secondary schools. The high student population in the area, along with the style of gallery, strongly influences the type of visitor. During the period of the trials it was the end of the school and University summer break.

The logs kept of each visitor's activities indicate that there were 155 users of the system during the 16 days of research over the 3 weeks. Of these, 7 can be deducted from a particular pair of users who exited and re-entered the system several time as a method of getting their bearings. Even if approximately 10% of sessions were due to gallery assistants re-use of the system, during consultation sessions, there were still around 133 valid users, of which 58 subjects returned completed questionnaires – around a 44% feedback rate.

Figure 4 shows the basic demographics of the sample population. The sample was predominantly female, and broad ranging in age from four years old to over 60. 50% of the subjects who answered the question about their occupation identified themselves as students with half (25% of total subjects) of those identifying their education as pre-A-Level. From the ages of the subjects, the total number of users under 16 is actually even higher (41%), suggesting that even more of the users were in full-time school education. From the other 50%

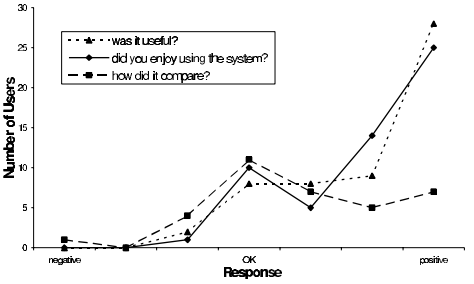


Fig. 5. User Satisfaction

of subjects, 9% of the total subjects were unemployed and 17% worked for City Art Gallery or other galleries.

The users of the system were reasonably typical of the normal visitors to the gallery, with a few exceptions. The gallery has a relatively high number of third age (over 60) visitors, few of whom were happy to use the system. In addition, the number of degree level students was lower than usual, due to the timing of the trials. Research by the Arts About Manchester project [3] into visitors to the City Art Gallery city site can be used to demonstrate strong similarities between our sample population and the population visiting City Art Gallery. Comparing the two populations they agree on the dominance of female visitors, a broad spread of age groups, predominantly young (often students), and far less third-agers. One noticeable difference is the larger number of family groups in our population, perhaps explained by the time of year.

Figure 5 shows the levels of user satisfaction recorded. Users gave a positive response to the system in terms of its usefulness, ease of use and comparison with other systems (where appropriate).

5 Discussion

The application described here makes use of some simple, well-understood techniques in order to deliver the required functionality. Our original requirements were better browsing support for the users; and better authoring support for the dynamic creation of browsing pathways.

The evaluation results suggest that browsing the hypertext generated by keyword linking was satisfactory. As opposed to the VOIR (Visualisation of Information Retrieval) system [11], we do not turn words above a frequency threshold into a hypertext anchor. Instead two anchors are associated with every artifact returned: the first returns a list of related artifacts; the second shows the most relevant unread artifact. Thus, we avoid the "over linking" problem whereby readers are overwhelmed by too many link choices. An alternative mechanism would be to filter anchors by some context-specific condition, but this adds to the complexity of the implementation. The returning of clusters of artifacts that

share common keywords, linked through a thesaurus, is a lightweight mechanism to support the search by analogy information seeking mechanism advocated by [19].

Better authoring support has been achieved by the creation of links purely through keyword associations. Dynamic link creation effectively removes the authoring task.

There are areas for improvement and further investigation.

5.1 Use of Thesaurus Structure

The similarity matching between object descriptions is achieved through the use of simple keywords. The structure in the thesaurus (broader, narrower and related terms), however, was not used for the purposes of this experiment, but could be used during the similarity calculations to improve the retrieval of "similar" objects. For instance, the fact that a tie and a scarf are both worn around the neck should lead to scarves and ties being more closely associated than, say a shoe and a scarf. Investigations are needed into how we might use the thesaurus relationships to calculate similarities, and whether such an approach provides significant improvement in the performance, or at least in the reported user satisfaction. The Semantic Hypermedia Architecture of Glamorgan [9] uses a thesaurus in this way to calculate similarities.

5.2 An Open Hypermedia Architecture

An Open Hypermedia architecture, such as the Distributed Links Server (DLS) [8] allows the integration of third party applications and documents into a system. The use of the DLS eases the process of constructing links through the use of link bases, which explicitly separate links from documents. Through the COHSE project, we are investigating the combination of an open hypermedia system with conceptual models (such as the thesaurus used in the annotation of collection objects).

5.3 Extending Coverage

The prototype is based on a small subset of items in the Platt collection. Currently, most electronic catalogue information in the gallery tends to be administrative rather than descriptive information. Extending the prototype will require the addition of more descriptive information in electronic form, which can be a labour intensive task. The successful reception of the prototype, however, suggests that this may be worthwhile.

Acknowledgements This work was supported by EPSRC grant GR/L71216. Thanks also to the members of Manchester City Art Galleries who were involved in the project, especially Liz Mitchell and the staff of the Platt Hall gallery for their cooperation. We would also like to thank Dr. Joe Bullock for his guidance on information seeking behaviour.

References

[1] Agosti, M., Gradenigo, G., and Marchetti, P. G., *A hypertext environment for interacting with large textual databases.* Information Processing & Management, 1992. **28**(3): pp. 371-387.

[2] Aitchison, J. and Gilchrist, A., *Thesaurus construction - A practical manual.* 2nd ed. 1987, London: Aslib.

[3] Arts about Manchester, `http://www.aam.org.uk/`.

[4] Bates, M., *How to Use Controlled Vocabularies More Effectively in Online Searching.* Online, 1988. **12**(November): pp. 45–56.

[5] Bates, M., *Indexing and Access for Digital Libraries and the Internet: Human Database and Domain Factors.* Journal of the American Society for Information Science, 1998. **49**(13): pp. 1185–1205.

[6] Bullock, J. and Goble, C., *TourisT: The Application of a Description Logic based Semantic Hypermedia System for Tourism*, in Hypertext '98, Pittsburgh. 1998.

[7] Bush, V., *As We May Think.* The Atlantic Monthly, July 1945.

[8] Carr, L., et al. *The Distributed Link Service: A Tool for Publishers, Authors and Readers.* World Wide Web Journal, 1995. **1**(1): pp. 647- 656.

[9] Cunliffe, D., Taylor, C., and Tudhope, D., *Query-based Navigation in Semantically Indexed Hypermedia*, in Hypertext '97, Southampton, UK. 1997.

[10] Furuta, R., et. al, *Hypertext Paths and the World-Wide Web: Experiences with Walden's Paths*, in Hypertext '97, Southampton, UK. 1997, pp. 167-176.

[11] Golovchinsky, G., *What the Query Told the Link: the integration of hypertext and information retrieval*, in Hypertext '97, Southampton, UK. 1997, pp. 67–74.

[12] Halasz, F. G., Moran, T. P., and Trigg, R. H., *Notecards in a Nutshell*, in Proc. of ACM CHI 87, Toronto, Canada,. 1987, ACM Press. pp. 45-52.

[13] ICOM Classification, `http://www.open.gov.uk/mdocassn/costume/vbt00e.htm`.

[14] Ingwersen, P., *Cognitive perspectives of information-retrieval interaction - elements of a cognitive IR theory.* Journal Of Documentation, 1996. **52**(1): pp. 3–50.

[15] Kheirbek, A. and Chiaramella, Y., *Integrating hypermedia and information retrieval with conceptual graphs*, in Proc. HIM '95 (Konstanz). 1995, Universittsverlag Konstanz. pp. 47–60.

[16] Lucarella, D. and Zanzi, A., *Information-retrieval from hypertext - an approach using plausible inference.* Information Processing & Management, 1993. **29**(3): pp. 299– 312.

[17] Marchionini, G., *Information Seeking in Electronic Environments. Cambridge Series on Human-Computer Interaction.* Vol. 9. 1995: Cambridge University Press.

[18] Nelson, T. H., *Literary Machines.* 1981.

[19] Pejtersen, A. M., *A Library System for Information Retrieval based on a Cognitive Task Analysis and Supported by an Icon-based Interface*, in Proc. of SIGIR '89, Cambridge, MA, USA,. 1989, pp. 40–47.

[20] Remde, J. R., Gomez, L. M., and Lanauer, T. K., *SuperBook: an automatic tool for information exploration-hypertext?*, in Hypertext '87, San Antonio, Texas, USA. 1987, pp. 175–188.

[21] Robertson, J., Merkus, E., and Ginige, A., *The Hypermedia Authoring Research Toolkit (HART)*, in ECHT'94, Edinburgh, UK. 1994, ACM Press. pp. 177–185.

[22] Saracevic, T., Spink, A., and M-M. Wu In , *Users and Intermediaries on Information Retrieval: what are they talking about?*, in User Modelling: Proc. of the 6th International Conference UM97. 1997, Springer: New York. pp. 43–54.

[23] Zloof, M. M., *Query-by-Example*, in Proc. of the National Computer Conference. 1975, AFIPS Press: Montvale, NJ. p. 431–438.

DIMS: Implementation of a Federated Information Management System for PRODNET II

Cesar Garita, Yasemin Ugur,Anne Frenkel,Hamideh Afsarmanesh,L.O.Hertzberger

University of Amsterdam, Faculty of Science
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{cesar, yasemin, annef, hamideh, bob}@wins.uva.nl

Abstract. The project PRODNET II[1] (Production Planning and Management in an Extended Enterprise) had as its main objective the development of a reference architecture and a support infrastructure for Industrial Virtual Enterprises (VEs). The PRODNET Distributed Information Management System (DIMS) supports the complex VE information management requirements and is based on a federated database architecture that has been specifically tailored and adapted to the specificities of the VE paradigm. This paper describes the kernel design, main internal components, and general implementation aspects of the DIMS system.

1 Introduction

The Virtual Enterprise (VE) concept can be briefly defined as an interoperable network of pre-existing enterprises that collaborate by means of specific IT components towards the achievement of a common goal. As a whole, these enterprises can function together and be regarded as a single organization for a determined period of time. Even though there are many definitions and ontologies around the VE paradigm, the fact is that any IT platform or infrastructure aimed at the support of these virtual organizations will certainly face an extremely complex and fractal-shaped problem domain. There already exist many software tools and standards that are able to cope with parts of the related interoperability issues, but there are a large number of challenges and open issues left unresolved. In terms of information technology, it is clear that advanced mechanisms must be designed and implemented to support the complex VE information management requirements [12].

In PRODNET, the Distributed Information Management System (DIMS) is the component that encapsulates all the functionality to support these requirements. In order to implement the DIMS component, two major pre-development phases were carried out in terms of the requirement analysis and the general system design (the

[1] This research was partially supported by the ESPRIT project-22647 PRODNET II involving the following partners: New University of Lisbon, University of Amsterdam, Estec, Uninova, Lichen, ProSTEP, Federal University of Santa Catarina, Miralago, Akros and CSIN.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 889–898, 2000.

results of these activities are documented in other papers [2],[3],[9]). In order to support all the information management requirements identified during the analysis phase, a federated database architecture was conceived during the design phase of the DIMS module. The design of the DIMS federated layer is based on the definition of a PCL (PRODNET Cooperation Layer) *integrated schema* that is represented and handled in all nodes. Data can be exchanged and shared through this integrated schema, but the proper access rights are defined locally at every enterprise in order to precisely specify the rights of external nodes on the local information of every node. Therefore, the DIMS properly preserves the federated information access and visibility constraints by means of well-determined export schema definitions. The general design of the DIMS has also been influenced by the PEER federated system architecture [1]. After this general design phase, which mostly focused on the specification of the federated schema integration approach, the internal DIMS kernel architecture itself was designed and implemented. Namely, the specific internal DIMS components were conceptualized, designed, and implemented in order to support the general federated schema architecture. The internal system design and final implementation of the DIMS kernel represent the main focus of this paper.

The rest of this paper is organized as follows. Section 2 describes the general DIMS reference architecture. Section 3 describes in details the main DIMS functional components. Finally, Section 4 summarizes the achieved results after the implementation of the DIMS module.

2 General DIMS Implementation Approach

In order to illustrate the role of the DIMS in the PRODNET architecture, it is necessary to first introduce the general PRODNET node architecture (see Fig. 1). This architecture has been extensively reported in other papers [4], and here only the basic elements are described. Every enterprise in the PRODNET II network of potential VE-members is considered as a *node* consisting of three major components:

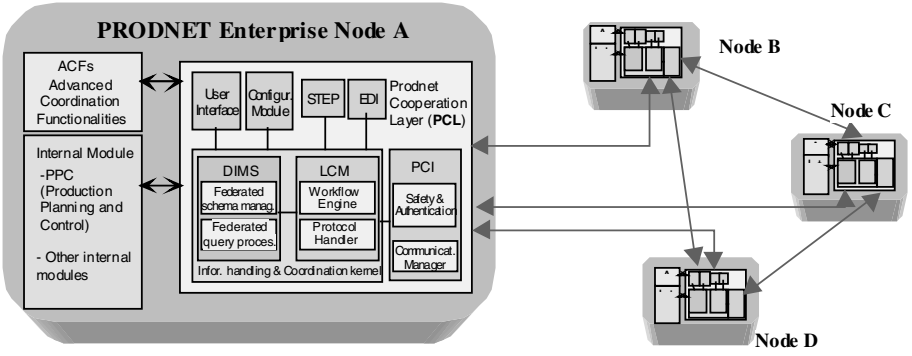


Fig. 1. Description of the PRODNET node architecture

an Internal Module, a PRODNET Cooperation Layer (PCL), and an Advanced VE Coordination Functionalities (ACFs) module. The Internal Module of a node basically consists of the internal information management systems of the company –such as its Production Planning and Control systems (PPC)-, necessary to accomplish its regular operations. The ACFs module provides some additional functionalities to extend the scope of the PCL, including the coordination of VE-related activities, and supporting tools for logistics operations. The Distributed Business Process Manager System (DBPMS) represents one of these ACFs.

Finally, the PCL component is responsible for the actual functionalities for the inter-operation between nodes in the PRODNET network. The PCL is the fundamental component that allows the enterprise to interoperate with others in the context of the VE. The PCL itself consists of several internal *components* [4]: (a) the Human Interface module; (b) the PCL Configurator; (c) the STEP and EDI modules; (d) the Local Coordination Module – LCM, which executes and controls the internal PCL workflow that determines the cooperation behavior of each PCL [5]; (e) the PRODNET Communication Interface (PCI), responsible for the actual communication channel among nodes in the VE network; and (e) the Distributed Information Management System (DIMS) that supports all the distributed information management requirements for the PCL operation. In the next paragraphs, the details of the DIMS reference architecture and its internal components are described.

The DIMS implementation infrastructure follows a three-tier architectural approach, also called client-agent-server architecture. In this case, the client tier is represented by all the other PCL components that request DIMS services via a DIMS client library. The agent (applications server) tier is represented by the DIMS Server Agent, together with the other DIMS internal operational components. This agent acts as a client of an Oracle database server, which in turn represents the server tier in this scenario. This section describes the main components of the DIMS applications server.

The general reference architecture of the agent tier embodies the following components, as depicted in Fig. 2: the DIMS Server Agent, the Federated Query Processor, the Export Schema Manager and Tool, the Internal DIMS Database Manager, and the DIMS Kernel Configurator. A brief description for every component is given in the paragraphs below:

- DIMS Server Agent: corresponds to the heart of the "agent tier" and is responsible for receiving and dispatching all the service requests issued by other PCL modules.
- DIMS Federated Query Processor (FQP): provides transparent access to data distributed over the nodes of the VE network, taking into account the specific visibility access rights (represented by export schemas) defined for every node.
- Export Schema Manager (ESM) and Tool (ESMT): ESM encloses the functionality to create and maintain the hierarchy of export schemas that are defined on the PCL local schema, based on the visibility access that are specified for a given node. The ESMT provides a user inteface to support the definition of the export schemas.
- Internal DIMS Database Manager: the DBMS that was used as "construction ground" for the DIMS is the Oracle DBMS (v. 7.3). It represents the server tier which provides all the functionalities that are expected from a DBMS (transaction management, data storage and retrieval, stored procedures, triggers, etc.).

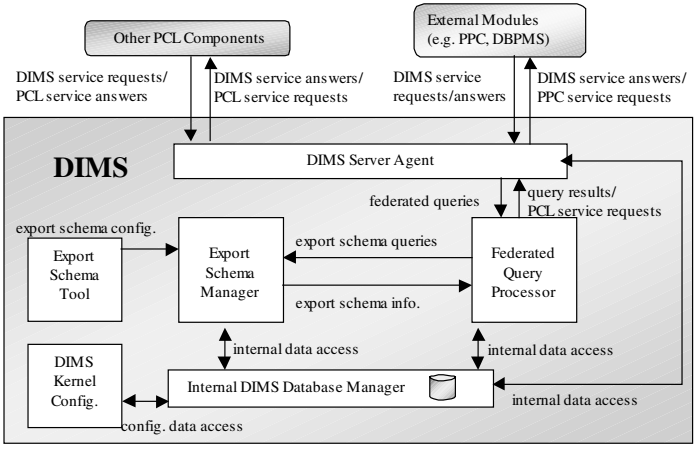


Fig. 2. General DIMS architecture approach

– DIMS Kernel Configurator: allows the specification of certain DIMS operation parameters such as communication port numbers and queries timeout durations.

3 DIMS internal implementation

This section addresses more specific design and implementation details regarding the DIMS Server Agent, the Federated Query Processor and the Export Schema Manager components of the DIMS architecture, that were introduced in Section 2.

3.1 The DIMS Server Agent

The Server Agent is the gateway to the internal DIMS architecture, encapsulating all the specific information management services for the PCL modules. The agent also provides a mechanism that allows internal DIMS components to reach the service interface of other PCL modules when required. To support this interoperation mechanism, both the DIMS and the other PCL modules are extended (wrapped) with a kind of interoperation layer, through which services can be reciprocally requested and answered. The interoperation layer is actually composed of two main parts (see Fig. 3): the PCL Module Interoperation Layer and the DIMS Interoperation Layer. Each of these layers is in turn decomposed into two major components: the client component and the server (or proxy) component. This subdivision is due to the fact that the interoperation between the PCL module and the DIMS is managed by a dual client-server interaction, in which each interoperation layer needs to simultaneously act as client and server of the other layer. For instance, the DIMS is able to request services from the PCL module (PM) via the PM client interface. The PM client in turn will contact the PM server that will carry out the service request. Similarly, the PM needs

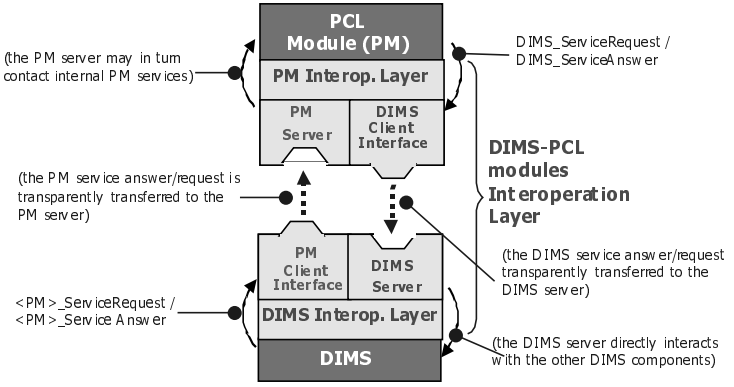


Fig. 3. General DIMS - PCL module interaction

to be able to request services from the DIMS via the DIMS client. In this way, the DIMS client will in turn contact the DIMS server (proxy) that will carry out the service request, as shown in Fig. 3.

All PMs and DIMS client interfaces are provided as DLLs that are linked to the corresponding main application. The DLL supports the interface to specific services that must be implemented in the associated server. Each client DLL provider must implement a mechanism in order to establish the communication with the corresponding server. For the implementation of the communication mechanism for each module, the PRODNET approach does not impose any constraints about it. In the case of DIMS, the implementation was done using remote procedure calls (RPC).

Furthermore, please notice that the communication mechanism to implement the functions provided in the client DLLs to request internal services, can be implemented either synchronous or asynchronous. In the synchronous approach, the requesting application program will not proceed with its execution until the request is fulfilled. The service request can also be satisfied asynchronously, which means that the issuing application will send the request and will be released to do other tasks while the service is carried out. Once the service request is accomplished, the answer is sent to the issuing application via a specific function. Both approaches can supported by the general PRODNET model, however in this document the asynchronous approach is assumed and described since it is the most commonly used approach.

In order to support the asynchronous approach, a pre-requisite for each PM (and the DIMS) is the implementation of an interface providing a pair of services required for the bilateral interoperation mechanism, namely a *ServiceRequest* and a *ServiceAnswer* function. These interface services are included in the client DLLs. For both of the request/answer functions, the parameters comply with a generic type definition that allows the transmission of elements of all the necessary types.

A basic interaction scenario of the general DIMS-PMs integration model using the service request/answer functions is also depicted in Fig. 3. For example, when the PM needs to request a DIMS service, it will asynchronously call the *ServiceRequest* function of the DIMS client interface. After the invocation, the PM will be released to continue its regular execution, and the request will be transparently transferred to the

DIMS server at the DIMS interoperation layer side. When the DIMS service request is fulfilled, the answer is sent to the PM via the *ServiceAnswer* function of the PM client DLL. This PM client interface will in turn seamlessly contact the PM server. It is also possible that the DIMS request a service from the PM in a similar way.

About the parameters of the service request/answer functions, they consist of three main predefined types: a *token* parameter, a list of *PCL parameters* of an abstract PCL parameter type, and a result condition parameter. The token parameter type supports the *context* definition for the execution of the service request, and specifies for instance a unique service request identifier, the identifier of the specific service being requested, and a timestamp, among other fields. The PCL parameters list allows the specification of the actual parameters that the specific module service demands. For this PCL parameters list, an abstract PCL data type has been defined from which a large set of specific data types can be derived and used in any module service.

3.2 DIMS Federated Query Processing

The PCL applications such as advanced VE coordination modules, and end users need to access VE-related data without worrying about the physical data distribution. At the same time, enterprises owning the information want to share different parts of the local data with different VE members and keep other part confidential. The Federated Query Processor (FQP) embodies the DIMS functionality to provide authorized access to proprietary VE-related data distributed over the VE network, depending on their visibility levels defined at the remote sites [8]. In this section, the main tasks of the FQP are detailed. Other approaches to the global query processing design and development in federated and multi-database systems can be found in [6],[10],and [11]. Most of these works address a query processing mechanism to support a general multi-database architecture. However, considering the VE peculiarities, these generic approaches should be tailored and extended such as has been done for DIMS.

The subtasks involved in the processing of federated queries in DIMS can be summarized as follows: when the query arrives at the DIMS, it is analyzed and decomposed into a set of single-site subqueries, each of which needs to be sent to only one site (VE node) to be processed. After that, the results of the sub-queries are gathered and merged into the final result. If necessary, the FQP interacts with the corresponding PPC to retrieve up-to-date local production data, during this process. More specifically, the main subtasks of FQP are depicted in Fig. 4 and described next:

– Query Reformulation and Decomposition. DIMS supports a set of high-level service functions to be used by the other modules. When one of these functions is called, FQP reformulates it into an internal query format using the parameters specified for every function. This reformulated query is then analyzed to determine the specific VE partners involved in the original query. Further, the query is decomposed into a set of simpler subqueries, so that each subquery involves the retrieval of data from only one VE node. Namely, each subquery needs to be sent to only one corresponding partner to be processed locally at that side.

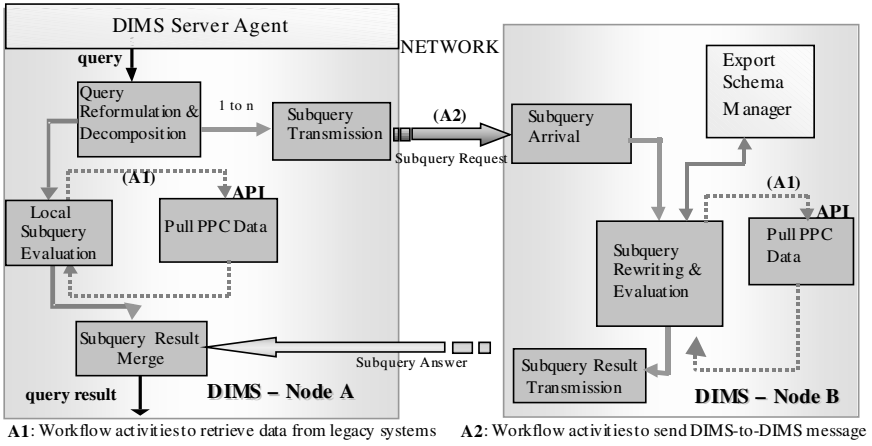


Fig. 4. Main FQP subtasks and interactions among VE nodes

- Subquery Transmission. This task sends subqueries to the necessary remote nodes. The subqueries, which are sent from one DIMS server to another DIMS server (in another node), comply with a specific format in order to facilitate the processing at the target node. A DIMS subquery request message format is composed by several fields including a type tag, target and origin nodes identifiers, VE identifier, and the query itself. To transmit the query from one DIMS node to another one, the DIMS exploits the workflow and communication facilities of the Local Coordinator Module and the PRODNET Communication Interface module, respectively.
- Local Subquery Rewriting and Evaluation. The PCL schema definition is the same in all nodes as described earlier, and any node can issue a query against it involving information from other nodes. However, the access rights of every node to the data that it can import from another node are precisely specified in the individual export schema defined for the origin node in the target node. Therefore, the arriving query will be carefully interpreted and evaluated by the FQP mechanism against the corresponding export schema of the sender to support the visibility access levels.
- Pull PPC Data. DIMS applications and end-users may need to get the most recently updated data from the local data sources inside the PPC system of the enterprise. To meet this need, DIMS communicates with PPC following the interoperability approach described in Section 3.1. The PPC functions invoked by DIMS through this mechanism allow the retrieval of data from the internal database system, and convert the result into the common data format expected by DIMS. The DIMS-PPC interaction is carried out using the *workflow* activities coordinated by LCM, which have been defined for data retrieval from legacy systems. Consequently, the DIMS gets the up-to-date data from the local production system through the PPC client interface and stores it in its internal database temporarily during the processing of the query. After this, the modified external subquery or local subquery is evaluated on the data stored temporarily in its internal database.
- SubQuery Result Transmission. This step is similar to the step of sending the subquery to the remote nodes, except that a different format of the inter-DIMS

message must be used. The first three fields of this DIMS subquery result message are defined as before in the "SubQuery Transmission" task. The fourth field corresponds to the result of the query, including the subquery identification, the return code of the subquery evaluation, and the content of the result itself.

- Subquery Result Merge. Once the subquery results arrive at the origin node which started the processing of the federated query being executed, the results of subqueries are kept in the database. When all the results have arrived, the merging step is achieved by a "union" operation of the individual results. The results are then sent to the requesting PCL module or end user interface. For a more detailed description of the query processing algorithm please see [8].

In terms of the interactions among the PCL modules to support FQP, it must be mentioned that the activity of sending/receiving Inter-DIMS messages is performed through both the workflow management mechanism provided by the LCM, and the communication means provided by the PCI module. With this strategy, the LCM workflow management mechanism is exploited to support flexible definition and changes in the process of sending/receiving DIMS to DIMS (enterprise to enterprise) messages depending on the business processes and procedures applied at every enterprise [5]. For a detailed description and examples of workflow definitions for FQP support involving several PCL modules and the PPC system, please see [8].

3.3 DIMS Federated Export Schema Manager

In the VE environment, every node must be able to grant different visibility levels and access rights on its local information to every other partner in every particular VE in which it is involved, based on the role that these nodes are going to play in the VE. To accomplish this objective following a federated database approach, every node can protect its autonomy and privacy by defining one detailed *individual export schema* based on its local schema, for every other node with which it shares information [7]. Furthermore, instead of allowing the enterprises to define individual export schemas on the local schema for every external "partner", we have generalized this basic idea to the definition of a complete *hierarchy of export schemas* based on the *role* of a given company in a VE. Namely, the decision of every node in the federation, regarding what part of its local information is going to be made available to the other nodes in the VE, will be based on the role that each of these nodes is going to play. Every partner of this enterprise in a given VE, will be associated with a role, and each role is related to an individual schema in the hierarchy of export schemas handled at this node. This hierarchy allows the grouping and classification of common export schema characteristics, facilitating the control and management of the individual export schema definitions. The main idea is to avoid the creation of an export schema for each one of the nodes involved in the VE, since every node may give the same access rights to two or more of these nodes.

For example, let us assume that for a VE there are three different kinds of roles that a given enterprise can play: the *coordinator*, *supervisor* (subordinated to coordinator), and *regular* VE partner (subordinated to supervisor). Clearly, for every role, different

information items must be made accessible from other nodes. For example, a VE coordinator needs to know information, which for a regular VE partner may even be a secret. For more details on the concept of role and export schema hierarchy, see [7].

In Fig. 5 the design of the database schema related with export schema management is presented. Through this schema, the recursive definition of elements of the export schema hierarchy and the role hierarchy are supported. For every different function (Role) that is going to be played in the VE, an external schema set (Export_Set) is defined, which at the end corresponds to the partner's export schema. Through the Export_Set, the proper visibility levels for the partners on the local schema of the enterprise are specified. An Export_Set can be either a single or a dependent export set depending if they are based on other export sets or not. With this approach, support for the general export schemas definition is provided, where not only the pre-defined export schema definitions at the level of VE, coordinators, supervisors, and partners, are considered, but also other hierarchies can be defined and supported as necessary. Also, an Export_Set consists of a set of schemas, which in turn can be single schema (EXP) or dependent schema (Dependent-EXP) following this definition strategy.

To operate on the described schema, an "Export Schema Manager" (ESM) module has been developed. The ESM is used to create a basic export schema, and then, to define dependent partner export schemas based on it. ESM is also used to create and maintain the hierarchy of roles. Furthermore, the ESM Tool (ESMT) is a graphical user-friendly application developed on top of the ESM that helps to define and create the export schemas, during the configuration phase of the VE [7].

In relation to the DIMS implementation environment and tools, the DIMS was implemented on Windows NT using Microsoft Visual C++, RPC and ODBC tools.

4 Conclusions

The implementation of the distributed/federated architecture of the DIMS in PRODNET, has proven to properly support the cooperative information sharing and

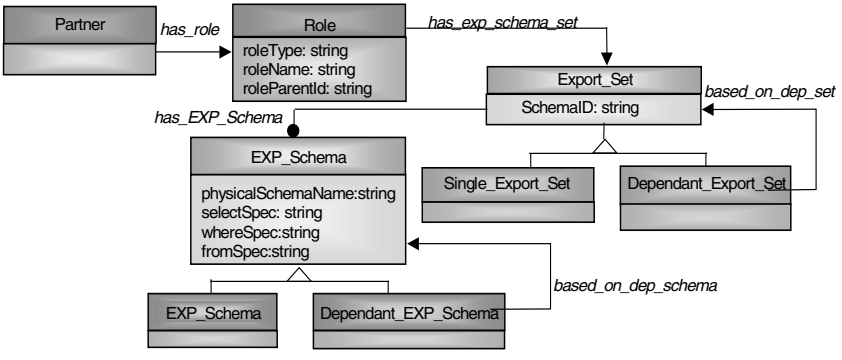


Fig. 5. Schema definitions for partner export schema management

exchange, node autonomy, information visibility levels and access rights for exchanged data among the VE nodes. The implemented DIMS server presents a three-tier architecture, which efficiently supports the interaction between the DIMS kernel and the other PCL modules. The DIMS Federated Query Processor provides access to the proprietary VE information for the authorized enterprises, while hiding the data location details from the end user. Furthermore, the DIMS Export Schema Manager and Tool properly support the definition of visibility levels and access rights for the information retrieval operations from other VE nodes. Finally, the implemented DIMS module satisfies all the information management requirements that were identified within the context of the PRODNET project, and provides a solid platform that can be extended in order to address future VE support enhancements to the current PCL implementation.

References

1. Afsarmanesh, H. et al.: Flexible and Dynamic Integration of Multiple Information Bases, Procs. of 5th Int. Conf. on Database and Expert Systems Applications-DEXA'94, LNCS 856, Springer Verlag, (1994) 744-753
2. Afsarmanesh, H., Garita, C., Ugur, Y., Frenkel, A., Hertzberger, L.O.: Federated Information Management Requirements for Virtual Enterprises. In: Camarinha-Matos L., Afsarmanesh H. (eds.): Infrastructures for Virtual Enterprises - Networking Industrial Enterprises, Kluwer Academic (1999)
3. Afsarmanesh, H., Garita, C., Ugur, Y., Frenkel, A., Hertzberger, L. O.: Design of the DIMS Architecture in PRODNET. In: Camarinha-Matos, L., Afsarmanesh H. (eds.): Infrastructures for Virtual Enterprises - Networking Industrial Enterprises, Kluwer Academic (1999)
4. Camarinha-Matos, L; Afsarmanesh, H.: The PRODNET Infrastructure. In: Camarinha-Matos L., Afsarmanesh H. (eds.): Infrastructures for Virtual Enterprises - Networking Industrial Enterprises, Kluwer Academic (1999)
5. Camarinha-Matos, L; Lima, C.P.: PRODNET Coordination Module. In: Camarinha-Matos, L., Afsarmanesh, H. (eds.): Infrastructures for Virtual Enterprises - Networking Industrial Enterprises, Kluwer Academic (1999)
6. Elmagarmid, A., Rusinkiewicz, M., Sheth, A.: Management Of Heterogeneous and Autonomous Database Systems, Morgan Kaufmann (1999)
7. Frenkel A, Afsarmanesh H, Garita C, Hertzberger LO.: Information Access Rights in Virtual Enterprises, in Proc. of 2nd Working Conf. on Infrastructures for Virtual Enterprises (2000)
8. Garita C, Afsarmanesh H, Ugur Y, Hertzberger L.O.: Federated Query Processing for Distributed Process Coordination in Virtual Enterprises. In: 4th Int. Conf. on Information Technology for Balance Automation Systems - BASYS'2000 (2000)
9. Garita, C., Afsarmanesh, H., Hertzberger, L.O.:The PRODNET Cooperative Information Management for Industrial Virtual Enterprises. Int. J. of Intelligent Manufacturing, (2000)
10. Jonscher, D., Dittrich, K.R.: An Approach For Building Secure Database Federations. Procs. of 20th Int. Conf. on Very Large Data Bases, (1994), 24-35
11. Meng, W., Yu C. T.:Principles of Database Query Processing for Advanced Applications, Morgan Kaufmann Publishers, (1998)
12. Silberschatz, A.; Zdonik, S.: Database Systems:Breaking out the Box. SIGMOD Record 26 (1997)

A Formal Treatment of a SACReD Protocol for Multidatabase Web Transactions

M. Younas[1], B. Eagelstone[1] and R. Holton[2]

[1]Department of Information Studies, University of Sheffield, UK
{M.Younas, B.Eaglestone}@Sheffield.ac.uk
[2]Department of Computing, University of Bradford, UK
DRW.Holton@scm.brad.ac.uk

Abstract: Issues of multidatabase transaction management within the Web are addressed. After examining the nature of the problem and reviewing current solutions, we argue that the classic ACID test of transaction correctness is not appropriate for Web transactions and propose new criteria based on SACReD properties. A new Web Transaction Model (WebTraM), based upon open and closed nested transaction models, is proposed and formally specified. Preliminary analysis demonstrates performance improvements over other Web transaction management methods.

1. Introduction

This paper proposes a new approach, WebTraM, to Web multidatabase transaction management (TM). WebTraM relaxes ACID (**a**tomic, **c**onsistent **i**ndependent, **d**urable) properties. Instead, it satisfies SACReD properties (**s**emantic **a**tomicity, **c**onsistency, **re**siliency, **d**urability,) which we argue are more appropriate to the Web environment. Advantages include: improved performance and resilience, support for heterogeneous autonomous systems, and advanced transactional applications.

2. Related Work

A review of Web TM is given in [16]. In summary, client-side implementations [3,9,10] has the advantage of direct interaction by clients with database systems, enabling use of DBMS TM. [3] extends ACID to include privacy and loyalty (i.e., "PLACID"), and addresses the unregulated nature of the Web by defining self-managed transactions where participant systems have no knowledge of each other. Heterogeneity of Web resources is resolved by interposing interfaces between Web TM and database systems, using transactional gateways and browser proxies [9], or Java applets [10]. However, the latter causes browser side recovery problems as they are prohibited from writing to local disk. The server-side TM solution, TIP [8, 6], uses 2PC but requires the TM of participating systems to comply with the TIP specification. Examples of middleware TM are [13,17]. Some middleware systems, [17], are extensible so that application specific solutions can be constructed, e.g. to provide concurrency control for extended transactions using plug-in codes.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 899–908, 2000.

A limitation of current Web TM is dependency on 2PC or its variant, *presumed abort* (PA). Consequently, resource blocking can occur, since subtransactions can not be unilaterally committed or aborted in prepare-to-commit state [14]. Also, excessive message overheads are incurred [12] that delays release of resources. Poor throughput can result, because on failure, transactions normally abort when not recoverable. The demand for 'prepare-to-commit' state limits 2PC-based protocols to homogeneous database systems [1]. Also, there is no support for advanced transactions requiring co-operation and data sharing among users [17]. In summary, ACID-based Web TM does not reflect the generic nature of Web, which provides opportunities for innovative complex forms of applications, accessing multiple heterogeneous autonomous information sources, with interaction and co-operation between users. The above deficiencies motivate our new approach, WebTraM.

3. WebTraM – An Overview

WebTraM supports multidatabase transactions on heterogeneous autonomously administered database systems. These are accessed via Web and database servers. TM manages transactions within the Web (*Web transactions,* denoted T_{web}*)* and within participating DBMS (*local transactions)*. Here, our concern is with Web transactions, since local TM is devolved to participating DBMS.

Informally, a Web transaction, T_{web}, is defined as *execution of a Web application which can be divided into well defined units that provides semantically correct transitions between consistent and possibly temporarily invalid states of the shared database systems.* This definition is based on that in [15], with qualification that states can be temporarily invalid when T_{web} partially commits.

A subtransaction, ST_i, of T_{web}, is *compensatable* if its effects on the database can be semantically undone by executing a compensating transaction. It is *replaceable* if there is an associated alternative transaction. Subtransactions can therefore be classified as *compensatable-only (CO), replaceable-only (RO), replaceable and compensatable (R-C),* or *non-compensatable and non-replaceable (NC-RC).*

T_{web} is defined formally (adapting the formalism in [5]) as a tuple, $T_{web} = (\boldsymbol{ST}, PaO)$, where $\boldsymbol{ST}$ is a set of subtransaction, $\boldsymbol{ST} = \{ST_i \mid i = 1...n\}$, and PaO is a partial ordering of the subtransactions which determines their order of execution. Each ST_i has a type (CO, RO, …) and is a sequence of operation, which may in turn be subtransactions, and which conclude with either abort or commit. ST_i is therefore defined by a tuple, $ST_i = (TST, Comp)$, where $TST \in \{CO, RO, R\text{-}C, NC\text{-}RC\}$, and Comp is of type seq (update | select | Comp | commit | abort) with the constraint that either commit or abort occurs only once within the sequence as the last element.

3.1 The SACReD Properties

Classical ACID properties are unenforceable within the Web-environment. Instead, WebTraM enforces *semantic atomicity* (SA) [16, 1], which allows unilateral commitment of subtransactions. SA requires that a transaction, T_{web}, commits only when all of its subtransactions, ST_i, have committed. Otherwise, T_{web} must abort, in which case the effects of any subtransaction that has committed must undone by

executing a compensating transaction. By enforcing SA intermediate states of local databases become accessible. Thus consistency can be enforced only at the component database level. Inter-database consistency cannot be enforced. In WebTraM, the shared state of databases temporarily remains inconsistent when any ST_i is locally committed and the global decision is to abort. However, such inconsistency is eventually removed using compensating transactions.

Durability requires that effects of a committed transaction must be made permanent in the respective databases even in the case of failures.

WebTraM defines *resiliency* as desirable, but not mandatory. This is important in Web environment, as transactions are vulnerable to failures due to unreliable nature of Internet, stronger requirements for local autonomy and consequentially increased likelihood of subtransaction failures. Resilience is increased by associating alternative transactions with subtransactions, that result in successful completion.

Thus in WebTraM, T_{web} is characterised by the properties of: *Semantic Atomicity, Consistency, Resiliency, Durability* (or *SACReD*).

3.2 The Architecture

The architecture within which WebTraM operates differentiates between *Web* and *database levels* (Figure 1). A transaction follows the commit procedure of open-nested transactions at Web level, and closed-nested transactions at database level. The architecture is based on the middleware approach [2] and comprises distributed components, including clients, HTTP/IIOP protocols, Web and database servers, a Web Transaction Coordinator (WTC), and Web Transaction Agents (WTAs).

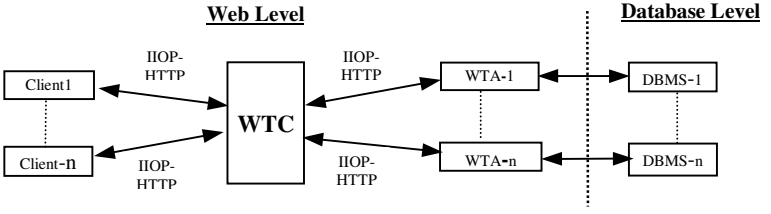


Figure 1. Architecture of the WebTraM

WebTraM protocols are implemented by WTC and WTA components such that autonomy of participating systems is preserved. It does not require special characteristics (e.g. prepare-to-commit state) from the participants, and therefore presents itself as just another user. WTC must offer the necessary functionalities needed by the transactions and works independently of participating systems. WTAs act as interfaces to underlying systems, and handle the protocol for a Web transaction.

4. Web Transaction Reliability Protocols

We formally define protocols used within WebTraM that ensure reliability [14] of a system. In the presence (and absence) of failures, a reliable system terminates a Web

transaction such that requirements of the SA property are met (to guarantee the consistency of the shared state of databases), and makes persistent effects of committed transactions (to ensure durability). The protocols must therefore deal with: t*ransaction failures* (F1) which occur if a transaction is aborted by participating DBMS [14]; s*ite failures* (F2), which occur if participating systems experience hardware or software failures; and *Internet failures* (F3) which give rise to communication failures resulting in lost or undeliverable messages.

We specify a protocol, P1, using CCS [11]. Initially we assume a *failure free environment* in which only transaction failures (F1) occur.

4.1 Failure Free Environment

P1 is defined as a multi-processes program in which WTC and each WTA are processes. WTC and WTA exchange messages during the execution of the protocol. First, we model individual behaviour of WTC and WTAs and then combine them to define the protocol P1. Message communication is diagrammatically shown in Figure 2, as a CCS state transition diagram.

Web Transaction Coordinator

Initially, a new transaction, T_{web}, is assigned to a WTC. When T_{web} starts, WTC records the begin of T_{web} in a log file using a simple write, and enters into a wait state, awaiting messages from the WTAs concerning completion of subtransactions. (Note that, at the start WTC does not send a prepare messages to WTAs, as is necessary in the IYV protocol [7]. Also, WTAs can send votes after processing ST, because WTC starts the protocol as soon as it starts the execution of a Web transaction).

$$\text{WTC} \overset{\text{def}}{=} \text{newTrans}(T_{web}) \,.\, \text{WTC}'(T_{web})$$

$$\text{WTC}'(T_{web}) \overset{\text{def}}{=} \overline{\text{s - write}}\ (\text{begin-of-}T_{web}) \,.\, \text{Wait}(0, \text{rec}, T_{web})$$

where rec = {1 .. k} is a set of indices of WTAs having sent their votes.

$$\text{Wait}(n, \text{rec}, T_{web}) \overset{\text{def}}{=} \textbf{if } n = \text{no-of-ST } \textbf{then } \text{WTC-commit}(T_{web})$$

$$\textbf{else} \sum_{k=1}^{\text{no-of-ST}} \text{vote}_k\,(\text{vote}) \,.\, \textbf{if } k \in \text{rec } \textbf{then } \text{ignore}\,(\text{vote})$$

$$\textbf{else if } \text{vote} = \text{local-aborted}_i \ \textbf{then } \overline{\text{s - write}}\ (\text{local-aborted}_i) \,.\, \text{WTC-abort}(ST_i, T_{web})$$

$$\textbf{else } \overline{\text{s - write}}\ (\text{local-commited}_i) \,.\, \text{Wait}(n + 1, \text{rec}, T_{web})$$

WTC first tests if the vote received from WTA_k, is duplicate, i.e., $k \in$ rec, in which case it is ignored. If $k \notin$ rec, then WTC acts according to WTC-commit(T_{web}) to commit T_{web}, or WTC-abort(ST_i, T_{web}) to abort T_{web} (if ST_i is not replaceable).

$$\text{WTC-commit}(T_{web}) \overset{\text{def}}{=} \overline{\text{f - write}}\ (\text{commit-decision}) \,.\, \text{send-commit}(0, \text{no-of-ST})$$

$$\text{send-commit}(i, \text{no-of-ST}) \overset{\text{def}}{=} \textbf{if } i = \text{no-of-ST } \textbf{then } \text{Global-commit}$$

$$\textbf{else} \sum_{k=1}^{\text{no-of-ST}} \overline{g - \text{commit}_k} \,.\, \text{send-commit}(i+1, \text{no-of-ST})$$

$$\text{Global-commit} \overset{\text{def}}{=} \overline{\text{Terminate}}\ (T_{web})\ .\ \text{WTC}$$

WTC forcibly writes the commit decision and sends global commit messages to all WTAs. It then terminates T_{web} according to Global-commit and starts the processing of a new Web transaction. Note that WTC does not require acknowledgements from WTAs regarding the commit of ST_i as they are already (unilaterally) committed.

$$\text{WTC-abort}(ST_i, T_{web}) = \textbf{if}\ \overline{\text{replaceable}}\ (ST_i)\ \textbf{then}\ \text{Wait}\ (n, \text{rec}, T_{web})$$
$$\textbf{else}\ \overline{\text{f - write}}\ (\text{abort-decision})\ .\ \text{send-abort}\ (0, \text{no-of-ST})$$

If an aborted subtransaction, ST_i, is replaceable, then it is replaced by an alternative subtransaction. WTC then waits for WTA's decision regarding commit or abort of the alternative subtransaction. If ST_i is not replaceable then WTC forcibly writes the abort decision and proceeds according to send-abort(i, no-of-ST) as follow:

$$\text{send-abort}\ (i, \text{no-of-ST}) \overset{\text{def}}{=} \textbf{if}\ i = \text{no-of-ST}\ \textbf{then}\ \text{Global-abort}$$
$$\textbf{else}\ \sum_{k=1}^{\text{no-of-ST}} \overline{g - \text{abort}_k}\ .\ \text{send-abort}\ (i+1, \text{no-of-ST})$$

$$\text{Global-abort} \overset{\text{def}}{=} \overline{\text{Terminate}}\ (T_{web})\ .\ \text{WTC}$$

WTC sends global-abort messages to all the WTAs, and terminates the transaction.

Web Transaction Agent

In P1, WTAs exhibit similar characteristics. Thus, we model a single generic WTA and then make use of the CCS re-labelling operator [] to define individual WTAs.

Initially, when a new ST_i is assigned to a WTA, it writes the begin-of ST_i in the log file using simple write operation and then starts the processing.

$$\text{WTA}(ST_i) \overset{\text{def}}{=} \overline{\text{s - write}}\ (\text{begin-of-}\ ST_i)\ .\ \text{WTA-process}\ (ST_i)$$

$$\text{WTA-process}(ST_i) \overset{\text{def}}{=} \overline{\text{execute}}(ST_i)\ .\ \text{Voting}\ (ST_i) + (\overline{\text{g-abort}}\ .\ \text{abort}\ (ST_i)\ .$$
$$\overline{\text{s - write}}\ (ST_i\text{-global-abort})\ .\ \overline{\text{Terminate}}\ (ST_i)\ .\ 0)$$

After executing ST_i, WTA sends a vote to WTC. Also WTA can receive a global-abort message from WTC amid the processing of ST_i. If that occurs, WTA must abort and terminate ST_i. This situation can arise when WTC receives an abort vote from another WTA, in which case WTC has to send global abort messages to all WTAs.

$$\text{Voting}(ST_i) \overset{\text{def}}{=} \textbf{if}\ \text{commit}(ST_i)\ \textbf{then}\ \text{local-committed}\ (ST_i)\ \textbf{else}\ \text{local-aborted}\ (ST_i)$$
$$\text{local-aborted}\ (ST_i) \overset{\text{def}}{=} \overline{\text{f - write}}\ (\text{abort-decision})\ .\ \overline{\text{vote}}\ (\text{abort})\ .\ \overline{\text{Terminate}}\ (\ ST_i)\ .\ 0$$

WTA forcibly writes the abort of ST_i, sends an abort vote to WTC, and declares ST_i as local-aborted. It then terminates ST_i , and stops processing.

$$\text{local-committed}(ST_i) \overset{\text{def}}{=} \overline{\text{f - write}}\ (\text{commit-decision}).\ \overline{\text{vote}}\ (\text{commit}).\text{Wait(WTC-decision)}$$

WTA forcibly writes a commit decision, sends a commit vote to WTC and then waits for WTC's decision.

$$\text{Wait(WTC-decision)} \overset{\text{def}}{=} (\text{g-commit} \,.\, \text{global-commit}(ST_i)) + (\text{g-abort} \,.\, \text{global-abort}\,(ST_i))$$

$$\text{global-commit}\,(ST_i) \overset{\text{def}}{=} \overline{\text{s - write}}\,(ST_i\text{-global-commit}) \,.\, \overline{\text{Terminate}}\,(ST_i) \,.\, 0$$

WTA simply writes the global commit of ST_i and changes the status of ST_i from local-committed to global-committed.

$$\text{global-abort}\,(ST_i) \overset{\text{def}}{=} \textbf{if}\ \text{local-committed}\,(ST_i)$$

$$\textbf{then}\ \text{compensate}(\text{comp}(ST_i)) \,.\, \overline{\text{s - write}}\,(ST_i\text{-compensated}) \,.\, \overline{\text{Terminate}}\,(ST_i) \,.\, 0$$

If ST_i is locally-committed and WTA receives an abort message, WTA must execute the compensating transaction for ST_i. This is logged by WTA by simply writing the compensation decision and then marking the end of ST_i.

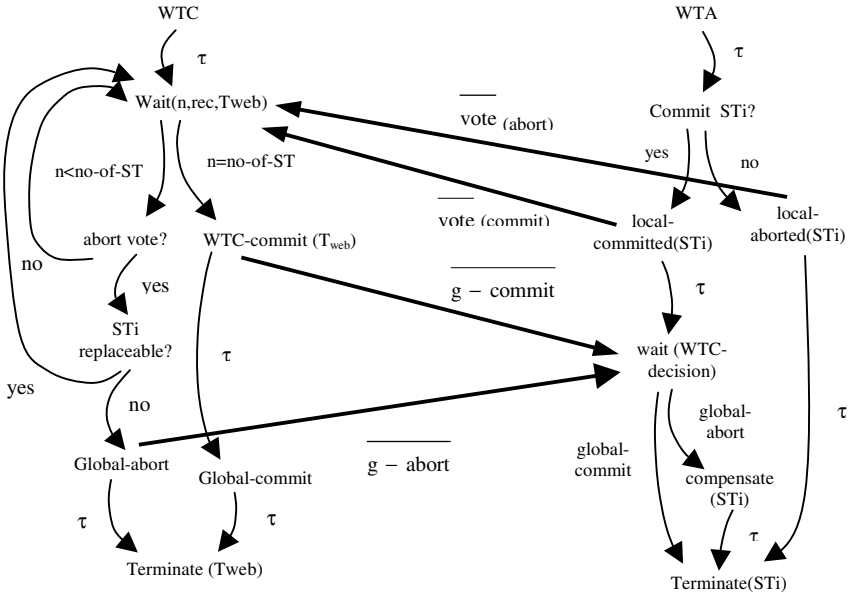


Figure 2. State Transition Diagram for P1 τ : internal action

Now we define the behaviours of individual WTAs as instances of the above generic WTA using the relabelling function [] as follow.

$$\text{WTA}_k\,(k=1..n) \overset{\text{def}}{=} \text{WTA}[\text{process}_k / \text{process}, \text{vote}_k / \text{vote}, \text{compensate}_k / \text{compensate}, \text{g-commit}_k / \text{g-commit}, \text{g-abort}_k / \text{g-abort}]$$

In the above, a label pair such as process_k / process defines the mapping of process to process_k, and vote to vote_k, and so on. We place the restriction that messages that can be communicated between WTC and WTAs are vote-commit, vote-abort, g-commit, and g-abort. This prevents floating of malicious messages between WTC and WTA.

$$L = \{ \bigcup_{i=1}^{\text{no-of-ST}} \text{vote}_i \} \cup \{\text{g-commit, g-abort}\}; \text{ L is a set of labels (or actions)}$$

P1 is therefore defined as follow:

$$P1 \stackrel{def}{=} (WTC \mid WTA_1 \mid WTA_2 \mid WTA_3 .... WTA_k) \setminus L$$

4.2 Failure and Recovery

We now describe how protocol P1 in the presence of F2 and F3 maintains the reliability of the Web TM system. We assume that an appropriate timeout period has been pre-determined according to the relation Time-out > Decision_T + Message_T, where Decision_T is the time WTC and WTAs take to reach a commit/abort decision, and Message_T is the time needed to receive a decision.

4.2.1 WTC Site Failure (F2)

WTC (sequentially) passes through the states: *initial, wait,* and *decision* (Figure. 2). The behaviour of WTC (Figure.3a) in these states is described as follow.

If WTC-state = initial $\wedge$ WTC-status = fail **then** WTC

else $\overline{\text{s - write}}$ (begin-of-T_{web}) . Wait (0, rec, T_{web})

When WTC recovers from failures in the initial state it must restart T_{web} from the beginning, as no information is logged about the beginning of T_{web}. If there is no failure then WTC changes to a Wait state.

If WTC-state = wait $\wedge$ WTC-status = fail **then** WTC-Restart

else WTC-commit (T_{web}) + WTC-abort (T_{web})

After recovery, WTC executes the restart process, WTC-Restart, and waits for votes of pending WTAs. If no failure occurs, it will commit or abort T_{web}, as appropriate.

If WTC-state = decision $\wedge$ WTC-status = fail **then** WTC-Restart **else** $\overline{\text{Terminate}}$ (T_{web})

If WTC fails after having written the global commit or abort decision, it will send again the decision messages to all WTAs, using the restart process as some messages may have been received by the WTAs, whereas others may not.

4.2.2 WTA Site Failure (F2)

WTA (sequentially) passes through the following states with respect to ST_i: *initial, processing, voting, wait* (Figure.2). The behaviour of WTA (as shown in Figure. 3b) in these states when errors of type F2 occur is described as follow.

If WTA-state = initial $\wedge$ WTA-status = failed **then** WTA (ST_i) **else** WTA-process (ST_i)

WTA-process (ST_i) $\stackrel{def}{=}$ execute (ST_i) **.** Voting (ST_i)

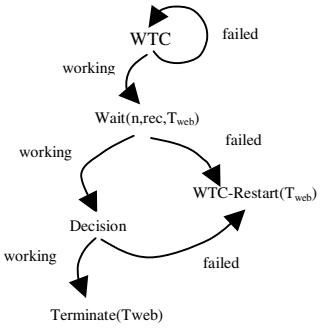


Figure 3a. State Transition Diagram of WTC during F2

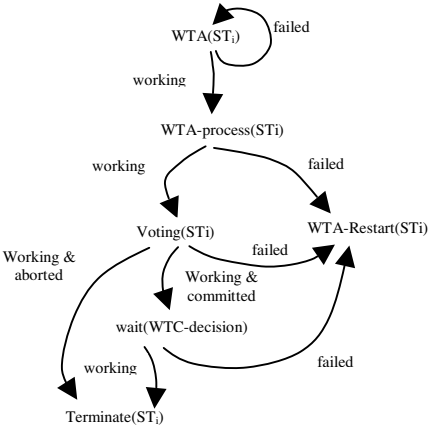


Figure 3b. State Transition Diagram of WTA during F2

WTA has to re-execute ST_i after recovering from failure, by following WTA (ST_i). Since ST_i has not yet started, there is no information available in the log file. In the case of no failure, WTA records the beginning of ST_i and starts processing.

If WTA-state = processing $\wedge$ WTA-status = fail **then** WTA-Restart **else** Voting (ST_i)

Voting (ST_i) $\stackrel{\text{def}}{=}$ **if** commit(ST_i) **then** local-committed (ST_i) **else** local-aborted (ST_i)

If WTA fails during the processing of ST_i, it has to follow the restart process, which checks the status of ST_i from the log file. On the other hand, if failure does not occur, WTA changes to the voting state.

If WTA-state = voting $\wedge$ WTA-status = fail **then** WTA-Restart
else Wait (WTC-decision) $\vee$ Terminate **.** 0

If WTA fails after writing the local commit or abort decision in the log file, but has not sent its vote to WTC, the following must happen. After recovery, WTA will follow the restart process, in which case, either WTA sends an abort vote to the WTC and terminate ST_i, or it sends a commit vote to WTC and wait for WTC's decision concerning global commit or abort.

If WTA-state = wait $\wedge$ WTA-status = fail **then** WTA-Restart **else** Terminate **.** 0

If WTA fails while waiting for WTC decision, i.e., to commit or abort ST_i, the following must occur. After recovery, WTA will query WTC to determine that global decision. This is done by following the restart process, because it is possible that WTC might have sent the decision while WTA was unable to operate due to failure.

4.2.3 The Internet Failure (F3)

Failures F3 affect the WTC and WTAs in the following states.

WTC Wait State

In this state WTC is waiting for WTAs' votes. If a vote from WTA fails to arrive due to F3, then WTC will timeout and will therefore assume ST_i has aborted. WTC will interpret this as a WTA site failure. The situation is modelled as follow:

$$\text{Wait}(n, rec, T_{web}) \overset{\text{def}}{=} \textbf{if } n = \text{no-of-ST } \textbf{then } \text{WTC-commit}(T_{web})$$
$$\textbf{else} \sum_{k=1}^{\text{no-of-ST}} \text{vote}_k(\text{vote}) \textbf{ . if } k \in rec \textbf{ then ignore}(\text{vote})$$
$$\textbf{else if } (\text{vote} = \text{local-aborted}_i \vee \text{timeout})$$
$$\textbf{then } \overline{s\text{-write}}(\text{local-aborted}_i) \textbf{ . } \text{WTC-abort}(ST_i, T_{web})$$
$$\textbf{else } \overline{s\text{-write}}(\text{local-commited}_i) \textbf{ . } \text{Wait}(n + 1, rec, T_{web})$$

WTC executes the action WTC-abort(ST_i, T_{web}) to abort ST_i if it receives an abort vote or if it times out due to lack of a response from WTA.

WTA Wait State

In this state WTA is waiting for WTC's decision regarding the global commit or abort. If WTA fails to receive the decision message, it will assume a WTC failure.

$$\text{Wait(WTC-decision)} \overset{\text{def}}{=} \textbf{If } \text{no-WTC-decision } \textbf{then } \text{timeout } \textbf{.} \text{ send-request } \textbf{.} \text{ Wait(WTC-decision)}$$

If no global commit or abort message has been received and the timeout period expires, then WTA re-sends a request to WTC regarding the global decision.

5. Conclusions and Future Work

We have shown that the ACID properties are over restrictive for Web TM and therefore presented a new approach, WebTraM, which ensures SACReD properties. We have formally defined WebTraM protocols. We believe that formal specifications are necessary, because of the complexity of the Web environment, and the importance of establishing correctness criteria against which to verify and validate.

Preliminary analysis (Table 1) shows P1 performs better than 2PC and PA, in term of: number of messages, information logging, and consumption of system resources (within a failure free environment, as in [12,7]).

Table 1: Comparison of P1 with 2PC and PA (commit/abort of a Web transaction)

	2PC						**PA**						**P1**					
	M-Sent		FW-Op		M-Rel		M-Sent		FW-Op		M-Rel		M-Sent		FW-Op		M-Rel	
	C	**A**	**C**	**A**	**C**	**A**	**C**	**A**	**C**	**A**	**C**	**A**	**C**	**A**	**C**	**A**	**C**	**A**
Cr/wtc	2	2	1	1	2	2	2	2	1	0	2	2	1	1	1	1	0	0
Pr/wta	2n	2n	2n	2n	n	n	2n	n	2n	n	n	N	n	n	n	n	0	0
Total	4n	4n	2n+1	2n+1	3n	3n	4n	3n	2n+1	n	3n	3n	2n	2n	n+1	n+1	0	0

C: commit case, **A**: abort case, **n**: no. of participants, **M-Sent**: messages sent by coordinator and participants, **M-Rel**: messages needed to release system resources, **FW-Op**: no. of forced-write operations.

Fewer messages and forced writes (in commit case) are required. Also, WebTraM does not require a message from WTC to release system resources. P1 enhances resilience, and supports advanced transactions, heterogeneous systems, and preserves their autonomy. We are currently implementing and evaluating WebTraM as prototype CORBA-compliant middleware.

References

[1] O.A. Bukhres, K. Elmagarmid "*Object-Oriented Multidatabase Systems: A Solution for Advanced Applications*" Prentice Hall, 1996.

[2] P. A. Berstein "*Middleware: A Model for Distributed System Services*" Communications of the ACM, Vol. 39, No. 2, February 1996, (86-98)

[3] D. Billard "*Transactional Services for the Internet*" Proc. of Int. Workshop on Web and Database (WebDB'98), Valencia, Spain, March 27-28, 1998.

[4] P.Chrysanthis, K.Ramamritham "*Synthesis of Extended Transaction Models using ACTA*" ACM TODS, Vol.19, No.3, Sept. 1994, (450-491)

[5] Sylvanus A. Ehikioya, K. Barker "*A Formal Specification Strategy for Electronic Commerce*" IDEAS, Montreal, Canada, August, 1997.

[6] K. Evans, J. Klein, J. Lyon "*Transaction Internet Protocol: Requirements and Supplemental Information*" Internet-Draft, October 1997.

[7] Yousef J. Al-Houmaily, P.K. Chrysanthis "*Two-Phase Commit in Gigabit-Networked Distributed Databases*" 8th Int. Conf. on Parallel & Distributed Computing Systems, Sept., 1995.

[8] J. Lyon, K. Evans, J. Klein, "*Transaction Internet Protocol: Version 3.0*" Internet-Draft, April 1998 (http://www.ietf.org/ids.by.wg/tip.html)

[9] M.C. Little, S.K. Shrivastava, S.J. Caughey, D.B. Ingham "*Constructing Reliable Web Applications using Atomic Actions"* 6th Int. WWW Conf., USA, April, 1997.

[10] M.C. Little, S.K. Shrivastava "*Java Transactions for the Internet"* Proc. of 4th USENIX Conference on OO Technologies and Systems, April 1998.

[11] Rubin Milner "*Communication and Concurrency*" C.A.R. Hoare Series Editor, Prentice Hall, International Series in Computer Science, 1989.

[12] C. Mohan, B. Lindsay, R. Obermarck "*Transaction Management in the R* Distributed Database Management System*" ACM TODS, 1986, (378-396)

[13] OMG "*CORBAservices: Common Object Service Specification*" November 1997 (http://www.omg.org/corba/csindx.htm)

[14] T.Ozsu, Patric,V."*Principles of Distributed Database Systems*" Prentice-Hall, 1991

[15] M.E. Rusinkiewicz, A.K. Elmagarmid, Y. Leu, W. Litwin *"Extending the Transaction Model to Capture more Meaning*" SIGMOD, March 1990, (3-7).

[16] M. Younas, B. Eagelstone, R. Holton "*A Review of Multidatabase Transactions on the Web: From the ACID to the SACReD"* British National Conference on Databases (BNCOD), Exeter, UK, July 3-5, Springer LINCS, 2000.

[17] J.Yang, G.E.Kaiser, "*JPernLite: An Extensible Transaction Server for the World Wide Web*" IEEE Transation on Knowledge & Data Engineering, 1999 (639-657).

Generic Architecture of Web Servers Supporting Cooperative Work

Jarogniew Rykowski, Waldemar Wieczerzycki

rykowski@kti.ae.poznan.pl, wiecz@kti.ae.poznan.pl
Department of Information Technology
The Pozna• University of Economics
Mansfelda 4, 60-854 Pozna•, Poland

Abstract. A new approach to building Web-based framework for CSCW applications is proposed. The framework is based on Resource Server, a multiversion Web server of three-tier architecture, composed of an interpreter of a query language as main interface of the server, specialized object-oriented database of information resources as an engine, and an XML wrapper as a gateway for data repositories. A particular emphasis is put on providing the server engine with basic transactional properties to support cooperative work: atomicity, consistency and durability, on one hand, and users' awareness and negotiations, on the other hand. The proposed Web server can be used as a flexible and scalable information exchange tool for cooperative work in the integrated processes of development, production and maintenance, as well as for advanced e-business applications.

1 Introduction

The WWW – by now the most popular service over the Internet – grows and evolves rapidly, from a simple, read-only data storage system, as it was a few years ago, to nowadays universal, distributed platform for information exchange. New Web-based applications with freely distributed data, end-users, servers, and clients, operating worldwide, are central topics of many research activities. At first glance, the Internet is an attractive base for a distributed computer system supporting cooperative work (CSCW), since the Internet is the most flexible network with the architecture that could support group activities to maximum extent.

There are several well-known and widely used Internet communication services: e-mail, chat, news, file and data transfer, remote computing, etc. Most of these services are well scalable to many application areas, from a local intranet server for a few people, to global systems like search services and public network portals for millions of users per day. Web servers can store and give access to data of any type and size, offering in addition some utilities, as client- and server-side computing and data formatting. However, looking more deeply, the Web is still not equipped with several mechanisms, useful or even necessary for building a cooperative environment.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 909-918, 2000.

First of all, the cooperative environment mentioned above should provide *durability* for all the data stored on servers and accessible to clients. Next, it should preserve *consistency* of data that have to be always accurate and mutually up-to-date. Finally, *atomicity* of groups of logically related operations would be required. Changes of values must be applied either for all the data involved in the group of operations, or, if it is not possible, they should be prohibited as long as all the data are fully accessible. Those properties are very important in case of collaborative design applications and collaborative software engineering applications. They are extremely important in case of collaborative e-business applications that support business transactions [1], which in a real life are always atomic, consistent and durable. Without these properties deployment of e-business applications would be very restricted. Notice, that the above three properties: atomicity, consistency, and durability – are basic features of database management systems (DBMS). To provide them so called *ACID transactions* are commonly used, which in case of DBMS have also the *isolation* property [2].

Up till now, there is no generic transaction support for Web services. Although durability for Web data is preserved, the two remaining required features – consistency and atomicity – are not provided. This is due to the data model used for the Web servers, where only read operations are permitted, while updating is done out-of-control, usually over a flat file system or a back-end database a WWW server takes its documents from. The only way to improve consistency is to stop a WWW service, update all its documents, and start it once again. Note, however, that this may introduce several errors for remote clients, as they cannot know in advance whether the "hanging" system is soon going to be updated or simply the network connection is broken.

In such situation, a natural solution could be to override the DBMS transaction mechanism and to model it at a higher level, for the whole Web server. It is even more motivated, because a transaction mechanism provided at this level would cover not only database operations, but also all operations in other data repositories (e.g., files).

The classical transaction mechanism, however, is not rich enough for distributed, Web-based, cooperative environment [3]. First of all, the isolation property is too restrictive, since in practice it excludes any form of collaboration, especially in a loosely distributed environment. Next, transaction integrity should be relaxed. Finally, to support real cooperation among members of a working group, at least two additional mechanisms should be provided. First, it should be possible for a transaction to involve a group of users rather than a single user [4] as for classical DBMSs. In cooperative environment, where transactions are long and produce many intermediate results which should be visible for other users, this is a group of users that decides together how to complete the transaction. Second, in case of access conflicts that occur during the execution of both a single transaction and a set of transactions, some negotiations should be provided [5, 6] that potentially can resolve such problems in a dynamic way, as a result of discussions undertaken by members of the working team.

To summarize, Web servers require transaction mechanisms to correctly support cooperation activities. However, to be effectively used in the Web environment, the transaction mechanism from the classical DBMS should be substantially extended. Speaking very informally, at least two new features are necessary: *cooperation sup-*

port inside transactions, and *negotiation support* among transactions whenever an access conflict is detected.

There are two main contributions of this paper. First, the paper proposes a new architecture for a Web server. This architecture is of three-tier type, and it is composed of a query language interpreter as the interface to the server, a specialized object-oriented database of resources as an engine, equipped additionally with transaction and user managers, and an XML wrapper as a gateway to data repositories. Second, the paper introduces so-called *teamwork transactions* that are modeled and managed at the Web server level and that override database transactions. Speaking briefly, a single teamwork transaction is assigned to the entire group of collaborating users, still preserving, however, the identity of individuals. The proposed transaction model is inspired by the natural perception, that a team of intensively cooperating users can be considered as a single virtual user, with more than one brain, and more than two hands operating on several keyboards. We also propose extensions to transaction management methods, with respect to the specificity of negotiations in the scope of a single transaction and a group of related transactions.

The rest of the paper is organized as follows. In Section 2 overall system architecture is presented, with particular care taken on functionality of the lower and the middle tiers, i.e., repository manager, and server engine. In Section 3, the concept of teamwork transaction is presented, including basic operations on teamwork transactions. In Section 4 main aspects of data management are pointed out that are related to resource distribution. Section 5 concludes the paper.

2 Overall System Architecture

We propose a new architecture for a multiversion *Resource Server* compliant with recently proposed three-tier architecture for Web services [7]. This architecture consists in using an interpreter of the proposed *RML query language* as the main gateway for the Resource Server, *Resource Server Engine* for managing versions of resources as well as meta-data [8], *Transaction Manager* to control users' transactions, *User Manager* to keep information about users, and *XML* [9] *Wrapper* as the gateway for data repositories (Fig. 1). The RML query language is used not only to read server's data, but also to define, manipulate, and control both data and their corresponding meta-data (data descriptions) as well as transactions and users. The data model for the Resource Server Engine is based on the MHD versioning model originally proposed for multiversion object-oriented databases [10]. The Repository Manager with the XML Wrapper allows storing both data and meta-data across many repositories, not restricted to only flat file system or a back-end database.

The lower tier is responsible for physical data access. This tier is composed of three sub-layers: the XML Wrapper, Cache layer, and Driver layer. The way this tier works is the following. A request from the middle tier (from the Resource Server Engine described below) is accepted by the XML Wrapper. Based on the request parameters, mainly resource identifier or unique name, a resource is identified. For this resource, a check is made if it is already wrapped. If so, another check is performed if the wrapped resource value is still valid. The latter action stands for local cache of re-

sources for the XML Wrapper. If the resource is properly wrapped, its value is returned to the middle tier. Note that the returned value is XML-based.

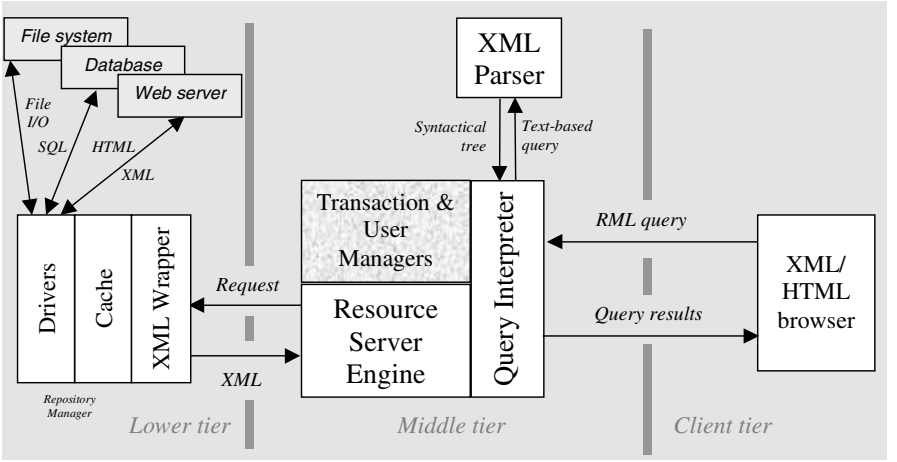


Fig. 1. The proposed architecture of queryable, multiversion Web server

If the resource is not wrapped, the Cache layer is contacted and a check is made if a raw resource value is already got from the external repository. If so, this value is sent back to the wrapper, where it is properly encoded into the XML, stored in local cache for the XML Wrapper and finally sent back to the middle tier.

If the raw resource value is not cached, it should be got from a given repository. Looking at repository type, proper driver from the Driver layer is contacted to fetch the value from given repository using respective URL. Although usually the repository types and real resource URLs are system-generated, users may freely change them. Moreover, different resources may be physically stored in different repositories. There are several possible drivers and repository types.

- *All-in-memory driver*: all resources are kept in the local operational memory, and in certain moments, all server contents is saved to a file;
- *File system driver*: information is loaded on-the-fly from XML-encoded files;
- *Other Resource Server*: information is loaded from another, remote Resource Server by the use of the RML query language;
- *LDAP and HTTP drivers*: information is loaded on-the-fly from LDAP or HTTP (WWW) server, respectively;
- *DMBS driver*: information is stored as XML-encoded tuples in a relational database.

As it can be seen, the main purpose of the lower tier is to access resources stored in different repositories and convert them into a common format. Once converted, resources are cached for future use. Note once again, that different resources may be physically stored in different repositories, enabling easy data distribution.

Middle tier is responsible for data accessing and searching. It is composed of five modules: Query Interpreter, XML Parser, Transaction Manager, User Manager, and Resource Server Engine. This tier works in the following way. A text-based query sent from the client-tier is passed to the Query Interpreter. The Interpreter invokes standard

XML parser to build syntactical tree for the query. This tree is passed back and traversed by the Query Interpreter according to semantic rules of the RML query language. The interpretation process is recursive, i.e., if intermediate results are queries, they are processed in the same way. Thus, values of some resources or just computed results could be used as parts of other queries. During query interpretation, values for resources are fetched from and stored into the Resource Server Engine. This module is responsible for management of resources, their versions and values, and their metadata. To this end, this module directly accesses the lower (physical) tier to read and write values of resources. The Resource Server Engine is implemented as an object-oriented multiversion database, with well-defined schema, and instances of objects fetched from the lower tier – the XML Wrapper.

The above-mentioned RML – Resource Manipulation Language – is XML-compliant, text-based language used for the following purposes: (1) maintaining definitions of the elements of the multiversion database, and (2) operating on these elements. On one hand RML queries are based on widely used SQL [11] programming language. On the other hand, queries are encapsulated into XML [12] document. Thus, universality of both database query language and main data-exchange language of the Internet is preserved, giving a new functionality: easy sending an SQL-like query via Internet to be interpreted by standard Internet software. This is due to the fact that RML queries and query results are XML-encoded texts. Thus, no special support is needed and any standard XML browser [13, 14], maybe embedded in a Java applet, could be used at the client tier.

The Transaction Manager and the User Manager modules from the middle tier are used for managing transactions invoked in distributed environment by users belonging to different workgroups. They are presented in details further in the paper.

3. Teamwork Transactions

3.1 Basic Concepts

Before we introduce teamwork transactions, we have to define some basic concepts. A Web Resource Server can be accessed in practice by an arbitrary number of users who work independently, or collaborate with others being members of the same working group. Depending on whether users collaborate or do not, and how tight is their collaboration, we distinguish two levels of users' grouping: conferences and teams.

A *conference* groups users who aim to achieve the common goal, e.g. to prepare a co-authored Web-site, design a new version of an electronic chip, write a common document. Users belonging to the same conference can communicate with each other and be informed about progress of common work by the use of typical Internet services, like e-mail, chat, and news. Conferences are logically independent, i.e. a user working in the scope of a single conference is not influenced by work being done in other conferences. It is possible for a single user to participate simultaneously in many conferences through different web browser windows, thus the intersection between two conferences need not be empty. In this case, however, actions performed in one conference are logically independent from actions performed in other conferences.

Users belonging to the same conference can collaborate tightly or loosely, depending on whether they perform the same common task, or they just aim to achieve the same final goal, respectively. Tightly collaborating users are grouped into the same *team* (i.e., a team is a subset of its corresponding conference).

A *teamwork transaction* is a flat, ordered set of operations performed by users of the same team, which is atomic, consistent and durable. In other words, a teamwork transaction is the only unit of communication between a virtual user representing members of a team, and the Resource Server of the architecture given in Section 2.

3.2 Isolation Levels

Two teamwork transactions from two different conferences behave in a way similar to classical database transactions, which means that they work in mutual isolation. In case of access conflicts, resulting from attempts to operate on the same resource in incompatible modes, one of transactions is suspended or delayed, depending on the concurrency control policy used in the Transaction Manager (cf. Fig. 1).

Two teamwork transactions from the same conference behave in a non-classical way, which means that the isolation property is partially relaxed for them. In case of access conflicts, so called *negotiation mechanism* is triggered, which informs users assigned to both transactions about the conflict, giving them details concerning operations which have caused it. Then, the users can consult their intended operations using conferencing Web tools, and negotiate on how to resolve their mutual problem. If commonly agreed, they can undertake one of the actions described in Section 4.2, in order to avoid access conflict; if they succeed, transactions can be continued.

A particular mechanism is used in case of conflicting operations of the same teamwork transaction, if different Internet users perform them. There is no isolation between operations of different users, however in this situation so called *notification mechanism* is triggered by the Resource Server, which aims to keep the users assigned to the same transaction aware of operations done by other users. After notification, users assigned to the same transaction continue their work, as if nothing happened. Notice that in case of users of the same team, we assume deep mutual confidence.

Different levels of isolation among operations performed by different users are illustrated in Fig. 2. Users U_1 and U_2 are associated to the same teamwork transaction, which is encompassed by conference C_1. Their operations are intermingled, but both users see all operations performed from the beginning of the transaction. If they modify the same resource, the more recent operation overwrites the previous one. Users U_1 and U_2 are totally isolated from users U_3 and U_4, whose transactions are encompassed by conference C_2. Users U_3 and U_4 are partially isolated, because they belong to different teams: T_{21} and T_{22}, respectively. U_3 does not see operations of U_4, except of the one that causes a conflict. In case of U_4 the situation is symmetric. This can help in solving access conflict in the near future by mutual negotiations.

As it has been already mentioned, in the proposed approach teamwork transactions override the transactional mechanisms at the *DBMS* level. It is feasible, since teamwork transactions are managed at a higher level (i.e. at a middle tier) than the data repositories, in particular the DBMS, do reside (i.e. at a lower tier; cf. Fig. 1).

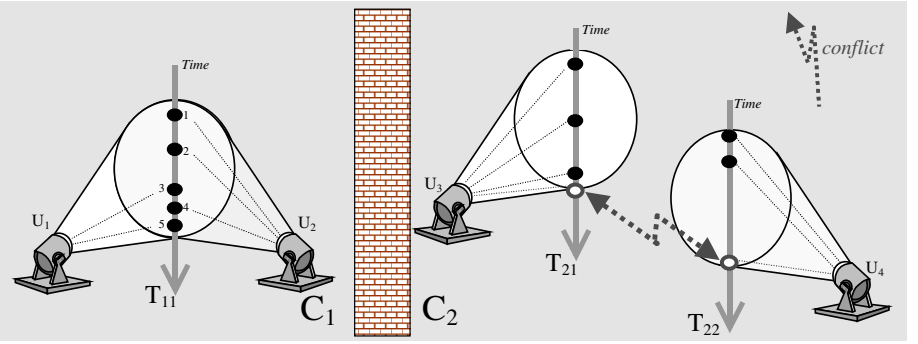


Fig. 2. Isolation levels among Internet users

In order to support users' notification and negotiations, a single conference that accesses DBMS resources is modeled by a single DBMS transaction. Thus, the DBMS transaction is potentially long-duration transaction, because it encompasses all DBMS operations performed in the scope of a respective conference. Notice that since in general many teams can work in the same conference and every team is assigned to a separate teamwork transaction – a single DBMS transaction encompasses potentially many teamwork transactions executed in the same conference.

4 Data Management

In this section we discuss possible distribution architectures by combining concepts introduced in Section 2 with teamwork transactions described in Section 3.

Due to the architecture of the Resource Server it is possible to share some resources among the servers, thus allowing resource distribution (cf. Section 2). A shared resource exists in several cached copies on several Resource Servers. Reading requests for cached copies are either performed locally, if the original is not changed, or forwarded to the Resource Server that keeps the original, if the latter has changed and a new, actual copy is needed somewhere. Writing requests to a copy are always immediately forwarded to its corresponding original, thus the global consistency of the original and its copies is always preserved.

Sample data distribution for a set of three Resource Servers *A*, *B*, and *C* is presented in Fig. 3. A client asks server *A* for a current value of resource *R*. This resource is cached from the server *B*. Thus, server *A* performs a RML query to verify cached value of this resource. If the value is not accurate, a new value in taken from server *B* by the query. Note that the whole process of caching is fully automated, and the user is not informed about the fact that the data he/she requested is physically taken from another remote server. Note also that the caching and access redirection may be performed at more than one level. Thus more than two Resource Servers are involved, for example server *B* may in turn redirect user access for resource *R* to server *C*. Of course, more far the original resource is away, longer it takes to verify and eventually get its value. However, regardless of a number of levels used, the obtained resource value is always up-to-date. This organization does not force the information to be localized and managed centrally, however, it makes it possible to keep all information

always consistent, not only from a point of view of a single user, but also from a point of view of the whole team.

Due to a fact that not all users are permitted to perform all tasks, a mechanism to verify access rights is provided, taking into account not only the users' rights, but also a place of the requested resource in the network of Resource Servers. Thus, from a point of view of a single Resource Server, two groups of users are considered: *local users*, controlled by this server, and *remote users*, controlled by other servers from the network. We assume that remote users should not have full access to all local resources. Instead, their access should be limited only to those resources whose values are explicitly marked as *released*, i.e., which are frozen and explicitly declared for public use. As a consequence, only local users have rights to update values of resources. Thus, a remote access never conflicts with a local one (and vice-versa), as all remote accesses concern only public, frozen, read-only resources.

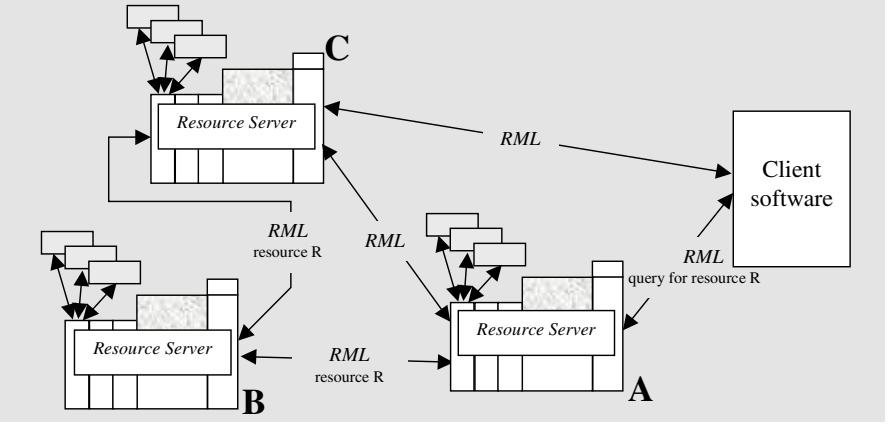


Fig. 3. Distribution of resources in a network of Resource Servers

From the point of view of a single Resource Server, to distinguish local and remote users, a special conference (cf. Section 3) named *public* is provided. The public conference concerns all remote users and released resources. There is only one such conference for a single Resource Server. A *local* conference concerns local users only and both working and released resources. There are as many local conferences as needed.

Thus, accessing released resources by members of the public conference is never conflicting with any other user. An access to local (i.e., not released) resources sometimes implies a conflict, and such conflict is negotiated inside a team working in a scope of a given local conference (cf. Section 4.2).

Access to resources through public and private conferences is presented in Fig. 4. The public conference C_p is reserved for the users accessing the Resource Server remotely. Note that only resources marked as "released" are accessible in such way. Two local conferences C_1 and C_2 are available for local users. The C_1 conference provides access to local (i.e., non-released) resources only, while the C_2 conference – to both local and released resources. Domains of local resources for C_1 and C_2 are not overlapping, thus users of these conferences are never conflicting. Notice, however,

that conflicts inside any of these conferences are still possible, although, they are detected by the Resource Server Engine and may be resolved by the negotiation mechanism. Notice also that users of neither C_1 nor C_2 conferences can fall into a conflict with remote users of C_p. In case of users of C_1, the reason is that they never access the released resources, while in case of users of C_2 – the reason is that they can simultaneously access with users of C_p only "released", read-only resources.

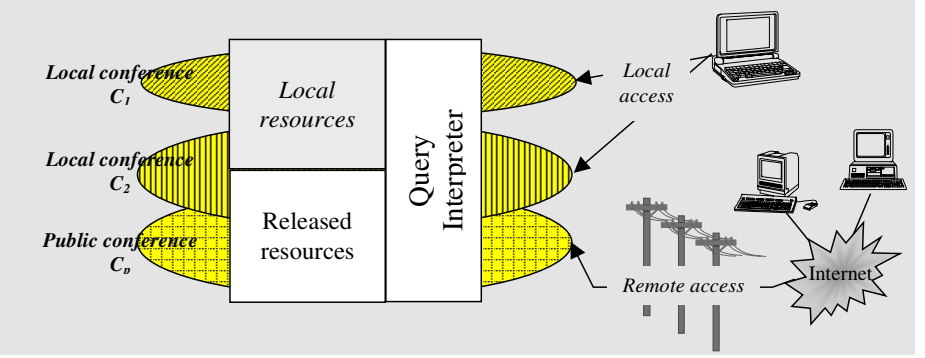


Fig. 4. Public and local access to resources

Note that the aforementioned discussion concerns possible access conflicts among transactions working in the scope of different conferences. The proposed approach assumes that in this case negotiation mechanisms are not needed and transaction are managed in a classical manner. However, the negotiation mechanisms augmented by users' awareness mechanisms become very useful while resolving conflicts among users working in the scope of the same conference. We discuss these topics in the next subsection.

5 Conclusions and future work

In this paper a new approach to build Web-based framework for CSCW applications is proposed. The framework is based on transactional and queryable Resource Server. The server if of three-tier architecture and is composed of XML Wrapper and Cache modules in the lower tier, and specialized, object-oriented database as the server engine in the middle tier. The engine is equipped with an interpreter of the RML query language. The RML language is used as the main gateway to the server, enabling creating, accessing searching, managing, and deleting resources from the server. The language is based on two standards: the SQL query language, widely used in the area of relational databases, and the XML language, being a commonly used standard in data exchange among Web servers and clients.

A particular approach to transaction management in Web-based systems supporting negotiations and collaboration has also been proposed. The approach is based on a new transaction model that is straightforward and natural, since strictly cooperating users are considered as a single virtual user. This assumption allows in practice unrestricted exchange of information among members of the same team.

The basic idea of the proposed approach is that collaborating users try to solve access conflicts detected during mutual negotiations. As a consequence, the problems related to concurrent execution of users' operations are resolved at the higher level than the level of the DBMS (as it happens classically). Speaking more technically, when access conflict occurs, all available information on this conflict is presented to the users. Then the users can negotiate, presenting they intentions concerning future work, and choose one of the proposed transaction management methods which aim at conflict avoidance.

A prototype of a Web server based on the proposed architecture is now being implemented. The implementation is based on Java programming language. Current work is concentrated on data distribution and intelligent caching of remote resources stored in different repositories. Future work goes towards building, on the framework presented in this paper, specialized applications for software engineering and automatic configuration of remotely executed Java programs, e.g., agents.

References

1. W. Cellary, W. Picard, and W. Wieczerzycki: Web-Based Business-to-Business Negotiation Support. Int. Conf. on Electronic Commerce TrEC-98, Hamburg, Germany (1998)
2. Elmagarmid A. (ed.): Database Transaction Models. Morgan Kaufmann (1992)
3. Nodine M., Zdonik S.: Cooperative transaction hierarchies: A transaction model to support design applications. The VLDB Conference (1984)
4. W. Wieczerzycki: Multiuser Transactions for Collaborative Database Applications. The 9th Int. Conf. on Database and Expert Systems Applications - DEXA'98, Vienna (1998)
5. Balasubramaniam R. and Tung B.: Negotiation in Network Based Requirements Analysis. The 32nd Hawaii Int. Conference on System Sciences – HICSS (1999)
6. Karacapilidis N., Papadias D., Pappis C.: Computer-Mediated Collaborative Decision Making: Theoretical and Implementation Issues. The 32nd Hawaii Int. Conference on System Sciences – HICSS (1999)
7. M.Carey, IBM Research, eds.: Post-Modern Database Systems: Databases Meet the Web. Berkeley University Course CS 294-7, http://db.cs.berkeley.edu/postmodern/
8. J. Rykowski: Architectures for Querying Contents of Web Servers. The 4th Int. conference on Business Information Systems (BIS 2000), Poznan, Poland (2000)
9. XML documentation: http://www.w3.org/TR/REC-xml-19980210
10. J. Rykowski, W. Cellary: Multiversion Databases - Support for Software Engineering. The 2nd Conference on Integrated Design & Process Technology, Austin, Texas, USA (1996)
11. Paul J.Fortier: SQL 3 - implementing the SQL Foundation Standard. At http://www.amazon.com
12. XML documentation: http://www.w3.org/TR/NOTE-sgml-xml-971215
13. IBM's XML parser for Java: http://www.ibm.com/developer/xml
14. Microsoft's XML parser for Java: http://msdn.microsoft.com/downloads/tools/xmlparser/xmldl.asp

Internet Search Technologies & XML

M.Montebello and R.Ciappara

Department of Computer Science and Artificial Intelligence
University of Malta, Malta.

Abstract

Searching and retrieving the right information from the World-Wide Web (WWW) has always been considered of foremost importance and of considerable A.I. intensivity. Internet search technologies have been evolving over the years and will continue to do so as the WWW will continue to expand in size and increase in popularity. In a desperate attempt to restore order to the WWW after the chaos that has developed due to its heterogeneous, unstructured and uncensored nature, the eXtended Markup Language (XML) is being heralded as the successor to HTML. In this paper we investigate the evolution of Internet search technologies and present a possible and viable solution in a functional system we developed and which makes use of XML at its very core. We discuss the design issues involved as well as practical issues such as tendencies and tactics employed by some of the major players in this well-sought area.

1 Introduction

The Internet is the World-Wide network of computers that began in the 1970s as a communications network between American government defence organisations. In 1984 control over the network was handed to the private sector and by 1989 the World-Wide Web (WWW), often referred to as the web, was developed. The WWW, which came out of work by the British scientists Berners-Lee et al. [BLCL+94] working at the European particle physics research establishment, CERN in Geneva, is the whole constellation of resources available on the Internet. The main idea is to merge the techniques of computer networking and hypertext into a powerful and easy to use global information system. Hypertext is text or graphics with links to further information, on the model of references in a scientific paper or cross-references in a dictionary. With electronic documents, these cross-references can be followed by a mouse-click, and in the case of the WWW, the staggering wealth of knowledge and experience, deposited and stored on server machines scattered across the Internet, is available to its on-line world.

When people access the web, they are either searching for specific information, or they are simply browsing, i.e. looking for something new or interesting (by a process often referred to as *surfing*). These two activities, searching and browsing, have associated applications over the WWW to access and help identify resources within the electronic documents respectively. The main difference

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 919–928, 2000.

between browsing and searching is that while no search statement or query is specified during the browsing activity, users explicitly express a search statement by submitting a query defining their information requirement to a web-search application, better known as a search engine. Links to potentially relevant information is presented by these search engines to the users, who can access and browse the information using a web browser.

2 Search Engines & MetaSearchers

The problem of browsing massive amounts of information in search of relevant material, described in the introduction, was initially addressed by the development and application of search engines and meta-search engines to locate information held on the WWW. In this section an in-depth analysis of these two search techniques is undertaken because of their direct relevance to the evolution process of Internet search technologies. Meta-searching is one technique employed in higher ordered systems discussed in Section 5, while search engines are also employed in the system we developed and present, apart from giving a major contribution to the meta-search technique itself.

It was pointed out earlier that the reason search engines were initially developed was to assist users in finding and accessing information on the WWW. They are still one of the primary ways that WWW users use to find information rapidly. Their interaction initialises when a user submits a query to one of the search engines. It scans through its index of site contents, which has been accumulated by cyclically crawling over the WWW, and displays the results in the form of an another web document which is presents to the user. This merely consists of a list of often thousands of links that are deemed relevant to the users' search query, and which the user has to scroll through to identify relevant information. The links the search engine returns are actual addresses of relevant sites on the WWW. These addresses are often referred to as Universal Resource Locator (URL) and are unique for each page available on the web. Apart from the URLs of relevant sites some search engines present also a summary about the site and what a user might expect. In this context, the term '*Search Engine*' is generally used to refer to the above technology that assists a user to find information. Numerous different search engines have been developed ([Sea]) and a full evaluation of them can be found in Jenkins et al. [JJBW98]. It will be beneficial here to discuss the difference between true search engines, directories and hybrid search engines by highlighting their specific characteristics in the next three sections. The reason for doing this is very important to draw out exactly what each system went out to achieve and in which architectural manner this was done. Similarly, meta-searching, even though not a search engine in its own right, will be discussed because it is extensively employed over the WWW and because it forms a vital part of some architectural set-ups of several systems.

2.1 True Search Engines

True search engines constantly visit web sites on the WWW in order to create catalogues automatically of web pages. Because these search engines run automatically and index so many web pages, they may often find information not listed in directories. Directories will be discussed in the next section, but only the fact that they are humanly administrated, can miss out on specific areas allowing gaps and loop holes due to oversights and inconsistencies. True search engines on the contrary can wend their way around the WWW by just following URLs listed in a web page and keep on crawling from one site to another, processing each page they go through and create their listings automatically.

The underlying architecture of true search engines has in general three major elements.

First is the *spider*, also called the crawler. The spider visits a web page, reads the whole page, and then follows links to other pages within the site. This is what it means when someone refers to a site being 'spidered' or 'crawled'. The spider returns to the site on a regular basis, such as every month or two, to look for changes. Everything the spider finds goes into the second part of a search engine, the index.

The *index*, sometimes called the catalogue, is like a giant book containing a copy of every web page that the spider finds. If a web page changes, then this book is updated with new information. Sometimes it can take a while for new pages or changes that the spider finds to be added to the index. Thus, a web page may have been 'spidered' but not yet 'indexed'. Until the information is indexed, that is added to the index, it is not available to those users employing the particular search engine.

Search engine *software* is the third part of a search engine. This is the program that sifts through the millions of pages recorded in the index to find matches to a search and rank the matches in order of what it believes is most relevant to the user's search statement.

2.2 Directories

Unlike true search engines, directories are created by humans. Sites must be submitted, then they are assigned to an appropriate category or categories. Because of the human role, directories can often provide better results than search engines. Directories are good if you are willing to sort through menu after menu, hunting for the best site. The problem with directories, as highlighted in the previous section, is that they are not fully comprehensive and might leave much to desire. Search engines are much better if you just want to see what is available on a topic since the engine does the hard work - crawling, sifting, indexing, as described in the previous section - and a user simply enters a query and reviews the resulting hits that are returned by the search engine. The ideal scenario would be if a mixture of the best qualities from the true search engines and the directories were merged together into one functional system, as described in the next section.

2.3 Hybrid Search Engines

To further confuse matters, some search engines also have an associated directory. These are sites that have been reviewed or rated. For the most part, these reviewed sites do not appear as the 'default' when a query is made to a hybrid search engine. Instead, a user must consciously choose to see the reviews. This is a case of merging the services provided by true search engines and directories into one service, as mentioned in the previous section, but the problem is that sites have to be submitted for review by the persons maintaining the directory. Reviewers often keep an eye on sites submitted, then choose to add those web pages that look appealing, in their opinion. These kind of search engines still inherit the problems directories have with the additional disadvantage that sites submitted for review to be included into the directory, are given no guarantee that they will be included. It is a matter of luck and quality and users are not given the choice to decide of their own free will what is relevant or not.

2.4 Meta-Searchers

Meta-search engines are the most recent step towards centralising searching on the WWW. Unlike search engines, they do not crawl the WWW themselves to build detailed indices, instead they take one query and send it to several major search engines. The results received are blended and aggregated into a single list and displayed. The MetaCrawler [SE95] and SavvySearch [DH96], for example, are two examples of academic meta-searchers that adopted this technique successfully in order to benefit from the pre-indexed information that can easily be accessed from the search engines' knowledge bases. A query submitted to one of these meta-searchers is automatically forwarded to a number of search engines which return a number of hits each. These hits are inspected by the meta-search engine and after removing duplicates, presents the user with the full list of hits.

2.5 Limitations

Different search engines offer different capabilities and vary in the way they perform their search, depending on the importance they give to different aspects of the retrieval process. Even though they may seem different, a number of similarities and common limitations can be identified about the facilities and capabilities they provide. These will now be discussed.

The most popular search engines are indexing agents that carry out a wide ranging, autonomous search of the WWW, keeping a record of the a site's URL depending on the information they extract from the site itself. The document title and document text are used to extract words to be able to position the site's reference in appropriate entries within an index the search engine generates, constructs and later utilises during queries. The WWW is constantly changing, so it's easy for search engine indices to become out-of-date. Different search engines update their index differently, like for example, AltaVista /citeAlt and Excite [Exca] indices may only be days to six weeks old, while others like Infoseek [Inf]

and Magellan [Excb] may be months old. Thus this means that when a user submits a query to a search engine, only the latest updated index is used. Considering that in the interval since the last index update fresh information might have been added to the web, the user is retrieving inaccurate and incomplete information. The time for a search engine's crawler to access the entire WWW is considerable, and will keep on increasing directly in proportion to the rate the WWW continues to grow ([Spe94,SRL96]), thereby amplifying this particular limitation.

As a direct consequence of this time lag some of the returned hits could have been removed from the WWW by the provider or the author resulting in a long wait by the user and eventually a broken link warning as the information is not available on-line anymore.

Even when the links still exist and are in fact on-line, it is an irritating and common experience for the querier to find access has been made to multiple entries for what is clearly the same page identified by several slightly different URLs. This is an increasing problem as providers realise that they must continuously alter their pages to keep some of the search engines visiting.

The indices that these crawling agents produce are not personalised or tailored to the user's actual needs and interests. This means, a query returns many false hits due to the limited expressiveness of the index terms, for example, a 'not' in-front of a term in a document negates its role. This will tend to increase as the WWW keeps expanding and flourishing ([EW95]).

The use of keywords, to describe exactly what the user is looking for, can be misleading and the hits generated can be referring to completely different matters within the alternative meanings of the same keywords specified. This is a well-known problem to natural language researchers who have been addressing it for some time. It boils down to the issue of homonyms and synonyms in documents ([GT97]), also alternative use of terms particularly jargon, when people are writing in non-native languages.

Meta-search engines attempt to use as much as search engines as possible, thereby standardising the query string to accommodate all the search engines utilised. This means that due to the simple query string utilised the meta-search engine does not make full use of all the capabilities and functionalities that the individual search engines have to offer.

Meta-search systems, have a higher recall record when compared with a single search engine because the top hits from a number of the major search engines are aggregated into one list. On the other hand, even though this high recall score, they have a low precision just like the search engines they depend upon. **Precision** measures how relevant the retrieved set of documents is to the user requirements [Kow97]. The results returned from search and meta-search engines are not focused enough, resulting in the users having to check through the documents returned and identify which ones are of interest to them, a sometimes tedious task.

The limitations outlined above helped to demonstrate that search engines may have been useful and beneficial with a limited WWW, but require much

improvement or further development to keep up with an ever growing WWW and the more demanding users. The use of search engines by meta-searchers might have solved the problem of low recall scores, but the precision scores still needed to be improved.

2.6 The Future of Search

The future of search technologies on the WWW is uncertain and hard to predict because of the ever increasing amount and diversity of information and users - the initial problem discussed in detail in the introduction. Search engines seem to be coping, but the WWW is estimated to contain anywhere from 400 to 500 million documents [FP98], and the major commercial search engines receive 15 to 20 thousand queries per minute. The amount of data on the WWW, the rapidity of change, and the hectic web of links pulling it together make truly coherent, comprehensive organisation nearly impossible. Researchers differ on where the problem lies - user interface, data collection and analysis, speed limitations of hardware, and so forth–but clearly, a new solution is required to safeguard the efficient and effective searching of the rich knowledge-base present on the WWW. Here is a sampling of new approaches from the information industry.

- *Mapping*
- *Collaborative Filtering*
- *Specialisation*
- *Client-Side Meta-searching*
- *Personal Agents*
- *Human Contribution*
- *Metadata*

3 XML - The Promise

In a desperate attempt to restore order to the WWW after the chaos that has developed due to its heterogeneous, unstructured and uncensored nature, XML [W3Ca] is being heralded as the successor to HTML. This will change the way web pages are currently authored, and is defined by the World-Wide Web Consortium (W3C) as "a common syntax for expressing structure in data". It became an official recommendation in February 1998. XML creates standardised tags which are located in a standard place thereby enabling XML-based search engines to find specific categories of information embedded deep in WWW sites. Current search engines allow users to search for just one item and they only inform the user if the item being searched for appears anywhere in the site in question. XML-based search engines will let users specify a major category plus other multiple sub-categories as well.

With XML in mind, the W3C set out to describe a framework in order to describe WWW resources - hence the new name Resource Description Framework (RDF), which is simply an XML application that delivers information

about information, or metadata. The RDF framework [W3Cb] can be used to describe information within a large WWW site by providing a site-map, thereby allowing end-users to quickly and easily find the information they need. RDF can also be used to describe all types of data, from WWW pages to local files and e-mail messages. Other potential expressions of RDF include search engine query results, database records, tables of contents and bookmarks - all in small information packets that could be pushed to a user's desktop. RDF could also become a standard mechanism for sharing information between WWW sites and ultimately between end-users.

The WC3 hopes that by providing a new framework, the search engines (among others) will be more receptive to metadata. Perhaps it will give them an incentive to reevaluate indexing more metadata, give support and adopt the new and useful tags.

Realistically, a meta tag framework has already existed for several years, complete with the comprehensive Dublin Core set, and yet nothing has happened. Therefore, it is hard to see search engines suddenly deciding that metadata is a priority. Add to this the fact that many at search engine companies do not trust metadata. It is fine to talk about how nice it would be if all web pages were categorised, but the companies know from experience that people will lie, mislead or do whatever they can to get on top.

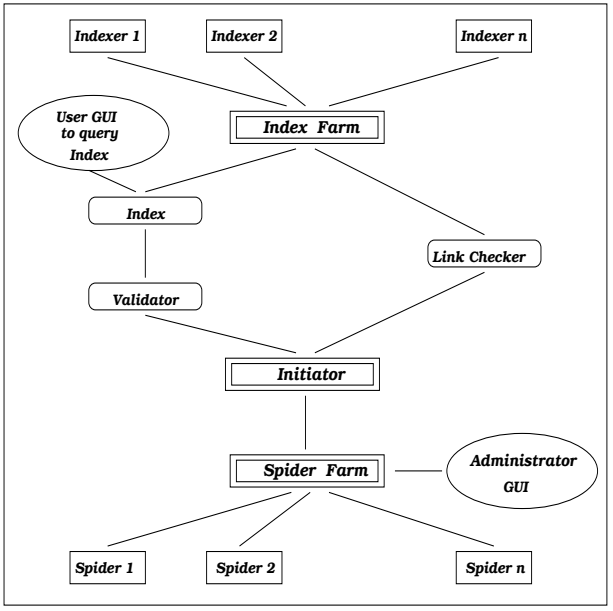
Standardised metadata seems to be the promise to a solution of a lack of machine-understandable semantics, one of the WWW's big problems. However, by using RDF metadata via XML, a WWW which is trustworthy can start to take shape with the help of W3C initiatives like DSig, P3P and PICS. Through automation, perhaps in the form of a system we present in the next section, the WWW can become a usable repository where the vast amounts of information that now, so often, seems to escape our reach, become manageable.

4 An XML Search Engine

The system we designed and developed takes full advantage of XML documents that are available over the WWW. It will function exactly like a true search engine with the exception that it will index those documents that have been XML authored only. Other HTML documents that will be encountered during its crawl will be utilised to obtain any links which may lead to actual XML data. The system is fully implemented in Java due to its ease in performing TCP/IP connections. The systems' architecture will now be presented together with an insight into the major components that make up our system.

4.1 Architecture

The major components of the system we designed and developed, namely, the *Initiator*, *Spider Farm* and *Indexer Farm* at the centre of the functional underlying application. Other components that contribute to the functionality of these



three major components are the *Index* itself, the *Link Checker*, and the *Validator*. Finally, two other components which concern the user interface itself are the *Administrator Interface* and the actual *Front-End* which a web user accesses to make use of the search engine's services. A more detailed description of the various components and the way they interact with each other follows:

- The Initiator: This component maintains a list of web page addresses (or URLs) which the system will download and parse. These URLs will eventually be passed to the Spiders through the Spider Farm. After that the system has been running for some time it will be receiving UTRLs from both the Validator and the Link Checker, both of which will have extracted these URLs from previously downloaded documents. One important task of the Initiator is to prioritise the URLs within its list. This is done by placing XML URLs coming from the Validator at the highest priority, then XML URLs coming from the Link Checker and finally those URLs that are simply HTML documents.
- Spider Farm: Spider Farm mainly manages the Spider threads that will be crawling the WWW. When a new spider is to be created, a request from Spider Farm to the Initiator is made and a URL is passed onto the new spider to access the actual document. The downloaded web page is returned to the Farm which passes it on to the Indexer Farm. Spider Farm obviously keeps track of all the Spiders that are active and reports this information to the Administrator's interface. More about this when the GUIs are expanded.
- Indexer Farm: This component has a similar function as the Spider Farm as it manages a group of threads, in this case *Indexers*. This Farm receives

downloaded document from Spider Farm and forwards them to one of the Indexers just created. An Indexer filters the document passed on to it from the Indexer Farm and extract URLs in the case of an HTML document, or indexes the contents in the case of an XML document. The resulting output will eventually be passes back to the Indexer Farm. URLs extracted are passed on to the Link Checker, while the actual indexes are passed on to the Index.

- The Index: All indexed documents will be recorder within the Index. The URL, date last indexed, indexed contents, etc... will all be stored within the systems database to be available when a user query is to be satisfied. The Index ensures in the case of XML document that the document is indexed accordingly to make effective use of the XML characteristics encountered.
- Link Checker: A record of all the links that have been spidered by the system together with the date when they were last spidered is held and employed by the Link Checker to ensure that all URLs coming from the Indexer Farm need to be spidered or not. This component also removes all those URLs whose extension is invalid. This is performed by comparing each URL extension with a list of over 2000 illegal extensions which will be ignored completely by our system.
- Validator: The Validator is a periodic component which goes through the whole index looking for documents which have been residing in the index over a set period of time. Those documents which pass this time threshold are resent to the Initiator for the system to update the index while ensuring that all indexed URLs are still live. The documents highlighted by the Validator go through the whole process discribed above as if they were new URLs.
- GUIs: There are two kinds of user interfaces which have been developed to accompany our system. The first one is an administrator GUI whose main purpose is to echo to the outside world what the system is doing. At the same time this interface is used to update any system settings like maximum number of spiders and indexers, time threshold employed by the Validator, etc... The second GUI is the one that a web user will access in order to query the XML search engine. This has the task of accepting a query from a user, parses the query, find any indexes that match the query and return the results to the user.

4.2 Current Implementation

All the components that have been discussed above have all been implemented and are functional at an evaluation stage. This is the case with both GUIs, as they are not yet well refined to be released to the general public, but at this stage their functionality is being met in order to perform the necessary tests of the system prototype.

5 Conclusion

Internet search technologies have been, are, and will continue to be a vital part of the WWW itself. Users depend upon them when utilising the web for any of their needs. The evolution of these technologies has been analysed in this paper, leaving an interesting question of how will the trend for future generations of search facilities will be. We argued in favour of employing the XML framework as a basis to develop a search engine which spawns spiders in search of XML documents. In this paper we presented the basic architecture of such a functional system and discused its major components. In future we will be performing in depth evaluation tests to analyse the effectiveness of our system and attempt to improve and optimise the information retrieved and indexed in order to make good use of the available WWW information resources.

References

[BLCL+94] T Berners-Lee, R Caillian, A Luotonen, H F Nielsen, and A Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, 1994.

[DH96] D Dreilinger and A E Howe. An information gathering agent for querying web search engines. Technical Report CS-96-111, Computer Science Department, Colorado State University, 1996.

[EW95] O Etzioni and D S Weld. Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert*, 1995.

[Exca] Excite Inc. *Excite*. http://www.excite.com/.

[Excb] Excite Inc. *Megallan Home Page*. http://magellan.excite.com/.

[FP98] R E Filman and S Pant. Searching the internet. *IEEE Internet Computing*, 1998. Special issueon Internet Search Technologies.

[GT97] V N Gudivada and S Tolety. A multiagent architecture for information retrieval on the world-wide web. *Proceedings of the 5th. RIAO conference Computer Assisted Information Searching on the Internet.*, 1997.

[Inf] Infoseek Corp. *Infoseek Home Page*. http://info.infoseek.com/.

[JJBW98] C Jenkins, M Jackson, P Burden, and J Wallis. Searching the www: an evaluation of available tools and methodologies. *Information and Software Technology*, 39(14-15):984–994, 1998.

[Kow97] G Kowalski. *Information Retrieval Systems: Theory and Implementation*. Kluwer Academic Publishers, Boston, 1997.

[SE95] E Selberg and O Etzioni. Multi-service search and comparison using the meta-crawler. *The Web Revolution. Proceedings of the 4th. international world wide web conference*, 1995.

[Sea] *Search Engine Watch*. http://search engine watch.internet.com/.

[Spe94] S Spetka. The TkWWW Robot: Beyond browsing. In *Proceedings of the 2nd. WWW conference '94: Mosaic and the Web*, 1994.

[SRL96] P Srinivasan, M E Ruiz, and W Lam. An investigation of indexing on the www. *ASIS '96 Annual Meeting of American Society for Information Science.*, 33:79–83, 1996.

[W3Ca] W3C. *Extended Markup Language*. http://www.w3.org/ XML/.

[W3Cb] W3C. *Resource Definition Framework*. http://www.w3.org/ RDF/.

Distributed Storage and Revocation in Digital Certificate Databases

Javier Lopez, Antonio Mana, Juan J. Ortega and Jose M. Troya

Computer Science Department, E.T.S. Ingenieria Informatica
Universidad de Malaga, 29071 - Malaga. SPAIN
{*jlm*, *amg*, *juanjose*, *troya*}@lcc.uma.es

Abstract. Public-key cryptography is fast becoming the foundation for those applications that require security and authentication in open networks. But the widespread use of a global public-key cryptosystem requires that public-key certificates are always available and up-to-date. Problems associated to digital certificates management, like storage, retrieval, maintenance, and, specially, revocation, require special procedures that ensure reliable features because of the critical significance of inaccuracies. Most of the existing systems use a Certificate Revocation List, a repository of certificates that have been revoked before their expiration date. The need to access CRLs in order to check certificate revocations becomes a performance handicap. Furthermore, they introduce a source of vulnerability in the whole security infrastructure, as it is impossible to produce a new CRL each time a revocation takes place. This paper introduces an alternative for the storage of digital certificates that avoids the use of CRLs. The system is designed to provide a distributed management of digital certificates by using Certification Authorities that, while being part of a whole Public-Key Infrastructure, operate over local certificates databases. Communication protocols between local databases have been designed to minimize network traffic without a lack of security and efficiency.

1 Introduction

With the rapid diffusion of the Internet in recent years transactions via networks are becoming increasingly common. In order to promote network transactions further, it is essential to ensure electronic authentication. Electronic authentication plays a very important role in assuring the reliability of network transactions and, indeed, the network itself. If electronic authentication is provided in an inappropriate manner, reliability is brought into question.

Therefore, a mechanism must be put in place for confirming the authentication of network users and the content of communications. Public key cryptography [1] makes such mechanism, the digital signature, possible. By using digital signatures we can obtain authentication in the form of digital certificates.

A *Certification Authority* (CA) is a trusted entity that issues certificates. The most basic certificate (identity certificate) binds the user's name (or identifier) to the corresponding public key, to which the CA's digital signature is affixed.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 929–938, 2000.

As the number of users in a system increases, the number of CAs increases too, conforming a *Public Key Infrastructure* (PKI). A PKI is the relying framework that allows a wide deployment of public-key technology as it provides essential reliability for electronic communication between users that cannot have a face-to-face relationship. Thus, by using a PKI, public key certificates management becomes possible, and a secure network environment can be established, enabling the use of security services (confidentiality, access control, integrity, authentication, and non-repudiation) for electronic transactions and for their supporting information technology applications.

A CA issues a certificate for a subject user such as an individual or a corporation, affirming something about the principal [2]. But, usually, the validity of this statement is limited in time; a certificate from my university that states that I teach Computer Security dated on 1993 will probably be useless to access the records of this year's students. In order to avoid this type of problems certificates usually include a validity interval. This does not solve the problems completely because there are circumstances when a certificate needs to be invalidated before the expected validity interval expires. For example, if the subject user loses the private key corresponding to the public key given on the certificate, or has it stolen or compromised, or if there is a possibility of this having occurred, the certificate has to be invalidated. The CA must publish the situation of revocation in a way that is accessible to the user and other parties involved through publicly accessible networks, such as the Internet.

This introduces the need of mechanisms to invalidate (revoke) the certificates before they expire [1]. Typically, CA makes the revocation information known by using a *Certificate Revocation List* (CRL).

Consequently, CRLs are basically repositories that identify certificates that have been withdrawn, cancelled, compromised, or should not be trusted for other specified reasons. Because a CA cannot force the destruction of all copies of a certificate, anyone who plans to rely on it must check it against a current CRL to ensure its validity. As a result, a CA must maintain continuity and promptness in the provision of revocation services, so that revoked certificates will not mislead users.

But all this process is not trivial; oppositely, it is complex because checking the validity of a certificate is not straightforward as the user must open a network connection to the issuing authority, find the CRL, and submit the certificate for checking. That is the reason why the issue of CRLs and the certificate revocation management are becoming an increasing focus of attention.

In this paper we introduce an alternative solution to the use of CRLs because we consider that they are not efficient for most applications. In section 2 we analyse the problem of using CRLs as mechanisms for revocation. In section

[1] A certificate has *expired* if its validity period has finished. Conventionally, a certificate that has not expired has been called valid. The previous discussion leads us to consider that this is not accurate and, consequently, we prefer to use the term fl certificate to refer to a certificate that has not expired. An active certificate is *valid* if it has not been revoked.

3 a new approach for revocation is introduced. This method is connected to the operation of Cert'eM, a hierarchical certification system based on electronic mail addresses that has been developed in our University. Section 4 presents two applications that use this system, and, finally, section 5 shows conclusions.

2 Certificate Revocation Lists

Most of PKI models use CRLs as the mechanism for certificate revocation. This method is defined by Recommendation X.509 [3], which is the recognized standard for public key certificate formats. In this Recommendation, a CRL is defined as a time-stamped list identifying revoked certificates, which is signed by a CA and made freely available. Each revoked certificate is identified in a CRL by its certificate serial number, generated by the issuing CA and included in the certificate.

The *pull method* of CRL distribution is the most common method that users employ to check the lists of revoked certificates. The CRL is not automatically distributed; on the contrary, users access to the list that is periodically published by the CA. But one drawback of this method is that the *time granularity* of revocation is limited to the CRL issue period [4]. As figure 1 shows, significant time intervals can take place since a user manifests the intention to revoke the certificate until the CA is informed of the fact, from that moment until the new list is issued, and from that moment until the CRL modification reach final users. During those periods, the risk of integrity in the system grows exponentially, and, certainly, it is not clear who holds responsibility in each of them.

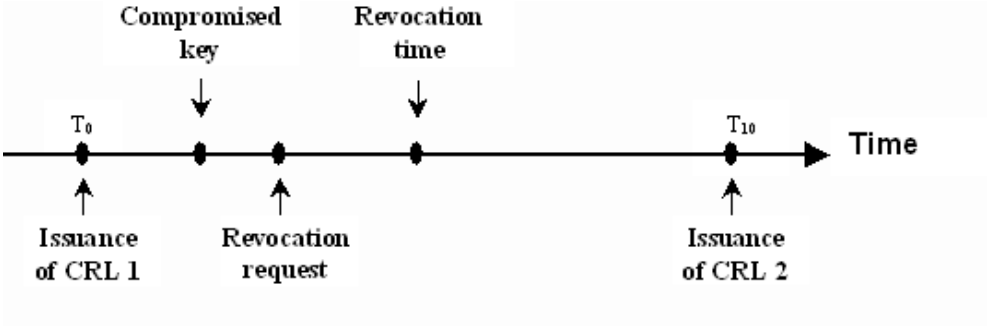


Fig. 1. Timeline of a certificate revocation process

Another limitation is the size that the CRL can reach. If the repository becomes very large, performance problems appear in terms of both communication overheads and processing overheads in certificate-using end entities. There are

two elements whose simultaneous growth can make a CRL difficult to be managed because of its dimension. The first one is the rate at which revocations occur, which is quite unpredictable, but that is clearly dependent on the size of the population of users covered. The second one is the certificates validity period, because once the CA introduces a certificate into the CRL, it is not deleted until the expiration date.

The first element - number of users - can not easily controlled, but the second one seems to represent a possible solution as it depends on the certification policy established inside the PKI. Thus, decreasing the certificate validity period during the certificate creation process, the CRL becomes smaller (very small validity periods could even eliminate the need to issue CRLs). But, in this case, what is positive for CRL size is negative for the general performance of the system. The reason is that the use of small validity periods implies that certificates must be issued more frequently. Therefore, this is a costly and ineffective solution and can be used only in some specific cases.

As modification of the validity period turns problematic, the solution for the uncontrolled growth of the CRL must be targeted by modifying the number of users. But, as this number cannot be decreased, the solution is to distribute users, that is, to partition the revocation repository into *CRL distribution points*, with each point containing a disjoint group of revoked certificates.

In version 3 of X.509 [5] the concept of distribution point is extended from previous versions, allowing those points to be established depending, not only on the subject type (final user or CA), but on the reason of revocation. Furthermore, certification policy allows each list to be issued at different times. The X.509 v.3 also introduces the concept of *incremental CRL*, or *Delta-CRL*. According to this concept, a CA does not need to issue new periodic versions of a CRL; it just needs to issue the modifications (Delta) from the last version.

Our consideration is that all these possibilities concerning the pull method represents a collection of too complicated solutions, and shows that this method is far from the solution that most of real PKI applications demand.

There is a less used method for CRL distribution, the *push method*. In this method, the CA sends the revocation lists to the users periodically, and does not introduce it into a repository as in the pull method. Such broadcasts are accomplished via protected communication means, such as secure e-mail or a protected transaction protocol. The major advantage of this approach is that important revocations can be distributed very quickly, without the time granularity delay problem inherent to the periodic revocation list approach.

However, there are two potential problems with this approach. Firstly, the requirement for a protected distribution method to ensure that CRLs reach the intended destinations. The protection of the distribution method represents an overload for the system. Secondly, the massive amount of traffic generated in order to notify all revocations. This problem could be solved if the broadcast is restricted; that is, if it is possible to establish, at the beginning, which revocations are broadcasted and who are the intended recipients. But this scheduling becomes impossible inside a large PKI.

Therefore, neither pull nor push methods are good options to solve generation, management and distribution of CRLs inside a PKI. We consider that the concept of CRL itself represents a drawback, and that the best solution is that one in which the knowledge of a revoked certificate is immediately available to users, but without a lack of performance in the system. This idea has been followed in the design of our certification system, Cert'eM, when certificate revocation problem has been faced.

3 Distributed Storage and Revocation

3.1 First Consideration: Certifying On-line

If a certificate is not included in a CRL then all the user knows is that the certificate was not revoked when the CRL was issued, but in most cases this is not enough. The reason for this is that CRL-based systems provide negative proofs (proofs of the negative validity but they give no evidence of the positive validity). Usually a positive proof of the validity of the certificate, or even better, a proof of the status of the certificate is needed; we call this proof a *validity statement* (VS).

As a consequence we affirm that, for uses other than historical (i.e. knowing that the certificated was once issued), the requirement of previous possession of the certificate does not represent any advantage to obtain confidence in the information of the certificate; on the contrary, it introduces serious obstacles to the certificate management and use procedures. We believe that an online certificate server can solve the certificate validity problem more efficiently than CRL-based systems (after all, the CRL retrieval requires an online request).

3.2 Second Consideration: Distributing Contents

Most of the systems that deal with the storage of digital certificates are based in centralized schemes. Some of them replicate the contents in different servers to distribute the requests among them, thus introducing synchronization problems.

Other systems do not provide storage for the certificates (this is done by the user) but do provide storage for the certificate revocations (using CRLs). These schemes, like those based in the X.509v3 standard, do not conform a distributed database (in this case, of certificate revocations) but separate and unrelated revocation lists. The CA establishes the revocation point at the time of issuance of the certificate and there is no standard way to balance the load between servers, or distributing the CRL among servers. Besides, the process of obtaining the appropriate CRL and all the subsequent Delta-CRLs just to know if an active certificate is still valid is quite complicated. As these CRLs contain the revocation of the certificates of many users, most of the information received in this case is completely useless for the requestor.

These solutions are very inefficient. The most efficient approach is to distribute the contents of the database of certificates among a series of servers

according to some established distribution criteria. A good distribution scheme should fulfil the following properties:

- An algorithm exists that applied to the known (key) data of the certificate (not the complete certificate), unambiguously identifies the server that contains it, and
- The scheme distributes the certificates in a balanced manner (i.e. the number of certificates stored in each server must be proportional to the capacity of the server).

3.3 Third Consideration: Distinguishing Names

One of the basic fields in almost all digital certificates is the *distinguished name* (DN). Sometimes, this DN is globally unique. For the purpose of using the DN in a distributed certificate storage scheme a globally unique DN is optimal for property one. A DN that is related to the logical location of the certificate is also desirable because it would help to fulfil the second property.

There are two possible schemes that are based in the use of DN. The first scheme is based in the *Domain Name System* (DNS) structure [6]. The recent DNS security extensions establish another proposal that allows authentication through digital signatures. Its name is *Secure-DNS* [7] [8]. These extensions describe a hierarchic PKI, integrated into the DNS database by adding a set of registers called *RR registers*. The public key of the CA of a zone is recorded in the *SIG RR register* and the public keys of the users of this domain are recorded in *KEY RR registers*, certified by the corresponding CA. But name servers expose several problems to store public keys because quite often DNS cannot be tightly coupled with its users, and, therefore, the link between real-world users and keys cannot be guaranteed (not conforming with article 8.2 in [9]).

The second scheme is the use of e-mail service structure as the base for the distribution of certificates in the different servers. This was the choice we took for the design of Cert'eM [10]. Cert'eM is a multi-hierarchical scheme that is based in the use of e-mail addresses as distinguished names and in the location of certificate distribution points in each e-mail office. The main element in the hierarchy is the *Keys Service Unit* (KSU), which integrates certification and management functions. Cert'eM proposes a scheme with various KSUs operating over disjoint groups of users, conforming a predefined hierarchy.

Figure 2 shows the system structure. The KSU hierarchy defined by Cert'eM is parallel to the hierarchy of Internet domains. The KSUs are associated to the corresponding e-mail offices.

Every KSU is managed by a CA (Figure 3). Additionally, it contains a portion of the certificates database to store the certified keys of its users. It must be emphasized that each user public key is stored exclusively in the database of his/her KSU. The third component is the key server, which receives requests and delivers the certificates. The key server manages a certificate proxy that keeps some of the recently received external certificates. The certified keys are

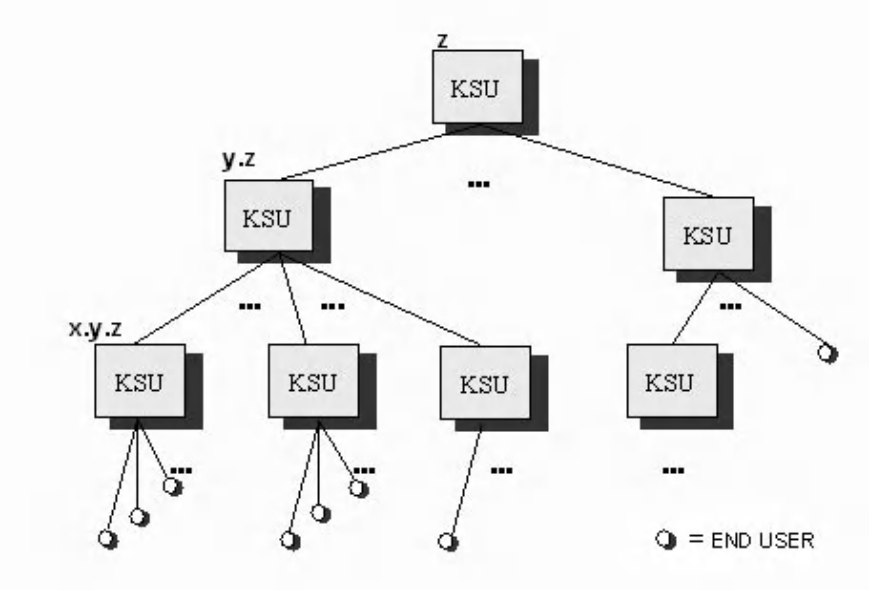


Fig. 2. Hierarchy of Cert'eM Nodes

managed solely by the corresponding CA; therefore, key updating and revocation are local operations that do not affect the rest of the system.

We want to emphasize that no CRL is used in the system. Instead the validation of certificates is achieved using the *Validity Statement* (VS), a timestamp statement signed by the CA attesting the status of the certificate at the time of issuance of the VS; therefore, in order to validate active certificates, the CA simply issues a VS.

To achieve a design that does not expose the mentioned problems of the use of CRL while still retaining their benefits, we impose the following restrictions:

- All the information related to the certification of a specific user must be located and managed at the corresponding KSU. Therefore, in case a CA decides to record certificate invalidation events, a *Local Invalidation Log* (LIL) can be managed locally. Notice that the LIL is completely different to CRLs. The LIL will be used exclusively by the CA for auditing purposes.
- Users must not distribute their certificates. On the contrary, the certificates must be kept in the database of the corresponding CA and distributed by their KSU.

When a user certificate needs to be invalidated (because his/her key has been lost or compromised, or because the CA has reasons to stop certifying the user) the CA simply deletes the certificate from its database and, if appropriate, stores

the revoked certificate in a LIL. This procedure is simple, immediate, requires no communication and can provide proofs of the certificate revocations in case the CA needs those proofs.

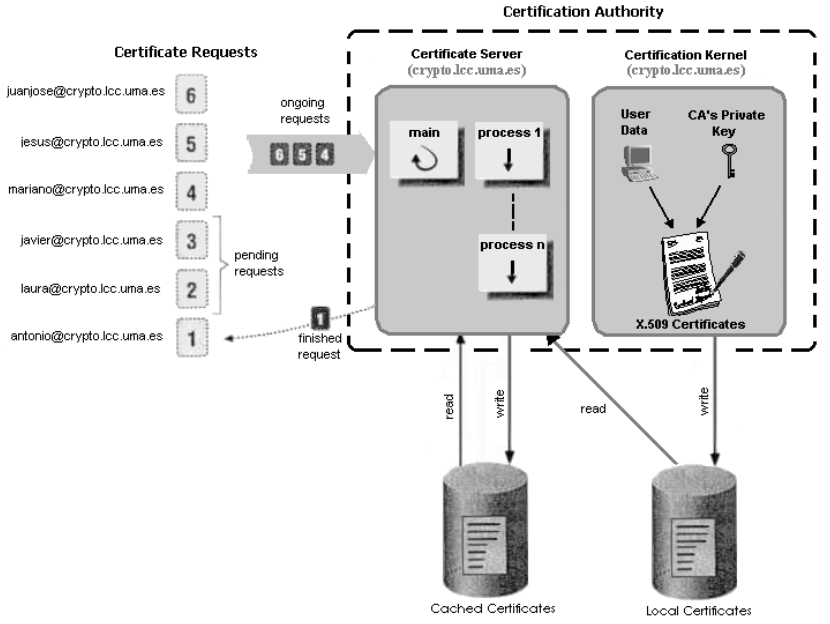


Fig. 3. KSU components

When the revocation takes place, existing active certificates are not useful any more because no VS will be issued to make them valid. The use of the VS prevents attacks based in old certificate reuse.

In case the key of a CA is changed, existing certificates are not useful any more and the CA must reissue all the certificates. Other systems need to notify users and request old certificates in order to re-certify their keys and distribute these new certificates. In our proposal there is no need to send new certificates and invalidate the previous ones, because all the certificates of the users of a KSU are kept in a local database. Thus, the change of the CA key (and hence, of the certificates issued by that CA) is transparent to users.

4 Example Applications

Two different applications have been used to test the system. The first application developed is used to secure the communication of sensible information

between the secretary of a university and the teachers. The purpose of this application is to test our certification and key distribution system in an environment that requires high security but that is not too large in terms of user population.

This application allows the teachers to receive, fill, and send back the official evaluation records of the students in the courses they teach. Teachers picks documents between those that are automatically selected for them according to the corresponding certificate. The document container (the java applet) enforces some steps to assist the teacher in filling the evaluations and to guarantee that the teacher information is error free, and provides in-transit security for the contents of the document. Cert'eM service is used to provide trustworthy authentication of both parts.

The second application is designed to make exams using Internet [**?**]. This application is used to test the generation of client-specific applets and the use of certificates in a bigger community of users distributed in different groups and locations. This case is also used to test Cert'eM in an environment that produces a high rate of certificate requests. Both applications are based in the dynamic creation of specialized Java applets that are responsible for the secure transport of the protected contents.

Figure 4 depicts a scheme of the dynamic applet creation process. The first step is not necessary in some cases. It is used, for example, in case of document sales, where this first negotiation phase is used to determine the terms and conditions (i.e. the contract) of the sale. This contract is processed by the applet generator to produce a specific sales agent for that contract and user. In other situations, there is no need to establish a contract and this phase is used to determine the security requirements that need to be included in the applet. This process needs access to trustworthy user identities and keys and is supported by the use of certificates that are managed, stored, distributed and revoked by Cert'eM servers.

5 Conclusions

Problems associated to digital certificates management, like storage, retrieval, maintenance, and, specially, revocation, require special procedures that ensure reliable features because of the critical significance of inaccuracies. Most of the existing systems use a Certificate Revocation List (CRL), a database of certificates that have been revoked before their expiration date.

In this paper we have introduced an alternative solution to the use of CRLs because we consider that they are not efficient for most applications. Moreover, we think that the concept of CRL itself represents a drawback, and that the best solution is that one in which the knowledge of a revoked certificate is immediately available to users without a lack of performance in the system.

This idea has been followed in the design of our certification system, Cert'eM, when certificate revocation problem has been faced. We have considered this

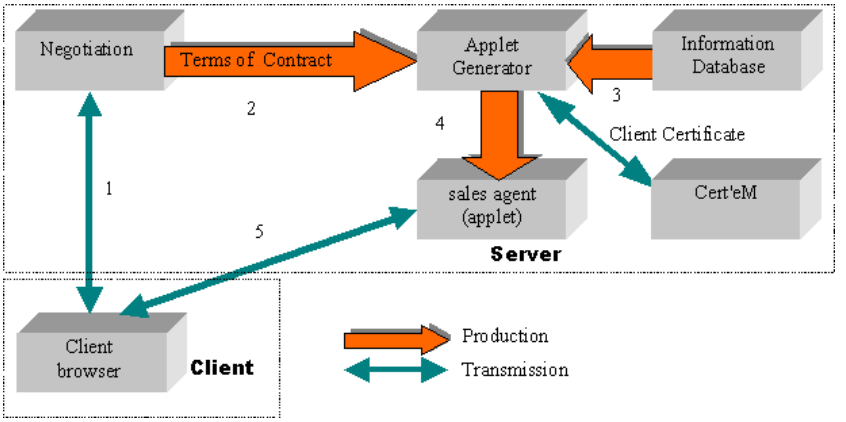


Fig. 4. Overview of the applet creation process

problem as priority in the design process, and consequently, have had a big influence in the rest of the PKI components.

References

1. W. Diffie, M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory. IT-22, n. 6. 1976, pp. 644-654.
2. Ilpf Working Group on Certification Authority Practices, "The Role of Certification Authorities in Consumer Transactions", Internet Law and Policy Forum, 1997.
3. ISO International Standard 9594, "Information Technology - Open Systems Interconnection Reference Model: The Directory", 1988.
4. W. Ford, M. Baum, "Secure Electronic Commerce", Prentice-Hall, 1997.
5. International Telecommunication Union, Itu-t recommendation x.509, "Information technology - Open Systems Interconnection - The Directory: Authentication Framework", 1997.
6. P. Mockapetris, "DNS Encoding of Network Names and Other Types", Request for Comment 1101, 1989.
7. D. Eastlake, C. Kaufman, "Domain Name System Security Extensions", Request for Comment 2065, 1997.
8. D. Eastlake, "Secure Domain Name System Dynamic Update", Request for Comment 2137, 1997.
9. European Commission, "Proposal for a European Parliament and Council Directive on a Common Framework for Electronic Signatures", COM(1998) 297 final, 1998.
10. J.Lopez, A. Mana, J. Ortega, J. M. Troya, "Cert'eM: Certification System Based on Electronic Mail Service Structure", Secure Networking - CQRE'99, LNCS 1740, Springer, 1999.
11. A. Mana, F. Villalba, J. Lopez, "Secure Examinations Through The Internet", Proceedings of Teleteaching'98, IFIP World Computer Congress, 1998.

Fine Grained Replication in Distributed Databases: A Taxonomy and Practical Considerations

Edgar Weippl and Wolfgang Essmayr

Software Competence Center Hagenberg, A-4232 Hagenberg, Austria,
{edgar.weippl|wolfgang.essmayr}@scch.at,
WWW home page: http://www.scch.at

Abstract. In order to design database replication scenarios in highly distributed environments it is necessary to investigate different options of replication. Based on this taxonomy of replication, we analyze how these types of replication can be implemented using Oracle8i. In the CommCoop[1] project we show how replication is used to support value-chain integration.

1 Introduction

Due to the Internet and the integration of subsidiary's networks to company-wide Intranets, distributed databases are increasingly common, even in smaller companies. Supply chain integration leads even further with different, independent companies sharing data, either in real-time or using overnight replication. As data quantities constantly increase, replication is very inefficient if the whole database (DB) is replicated but only a small part of the replica's data is actually accessed.

Another important issue is security. If independent companies cooperate, they do not want to share all data. Despite being tightly integrated into the supply chain, a supplier might also work for a competitor and should therefore be denied access to confidential data.

In this paper, we start by looking at different combinations of how data in database systems (DBS), similar to federated DBS [8], can be replicated unconditionally. Before starting to implement highly distributed DBS, one has to analyze the requirements according to this taxonomy of replication. Furthermore, consideration has to be given of how and when data should be duplicated in other DBs. As some DBs may use dial up connections to replicate, global transactions are not possible. We continue by presenting how these different replication schemata can be implemented using Oracle8i. Finally, we will give an overview of how these ideas have been used in a worldwide distributed environment.

[1] This work has been funded by the KPlus program of the Austrian government, the province of Upper-Austria, and the Chamber of Commerce of Upper-Austria.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 939–948, 2000.

2 A World-Wide Replication Environment

Within the project CommCoop (Communication Platform for Cooperation of Manufacturing Machines), a joint effort with our industrial partner AMS Engineering, we develop a system to integrate intra- and interregional DBs of manufacturing machines for problem analysis and semi-automatic problem solution.

AMS Engineering is a company that designs, manufactures, and installs machine tools that are controlled by a computer. These computers have access to mostly local DBs and read outstanding orders to be processed from the DB as well as write production data, like errors, supply shortages, etc., into the DB.

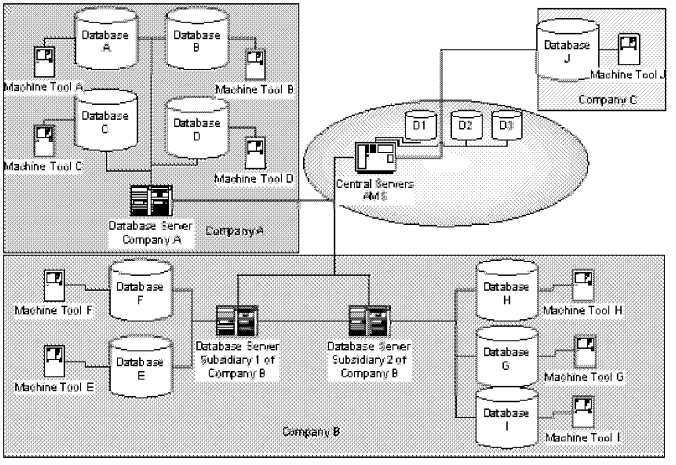


Fig. 1: An example of a worldwide manufacturing environment.

Fig. 1 shows three of AMS' clients: Company A, B and C. Company A has four machines. Each machine is connected to a DB. The DBs are connected to a central, company wide DB server. Company B has two subsidiaries, both being similar to company A. Machine E and F are connected to their local DB as well as to the primary subsidiary DB server. The subsidiary DB servers are connected to each other so that each subsidiary can access some data of its counterpart. Company C is a small company with one machine and its local DB, only. All companies are - at least temporarily - connected to AMS' federated site. Parts of the company data, like machine failures, maintenance activities, etc., are mirrored at central DBs D1 to D3. Each machine being connected to a local DB stores data in this DB while operating. If the machine fails, it saves its parameters describing the error conditions, too. Once repaired, the steps that have been necessary to correct the malfunction are recorded in the DB.

The machine tools produce complex parts like sockets and hinges. Each machine consists of one or more modules, with one or more units in each module. AMS' clients vary in size: There are companies that have only one machine in-

stalled, while others have up to sixty machines in one or more plants. Depending upon how many machines are connected to one DB and the machines' speed, up to four insert-operations per second have to be made.

The project's goal is to provide means that in case of a machine failure, one can submit a query to find out whether similar situations have happened before and what was done to restart the machine. Optimizing these queries which can be submitted anywhere in the network is not the only issue we have to deal with. As companies do not want to publish data on when they produced which products, not all data can be replicated throughout the entire net (security). Furthermore, connection speeds and bandwidth may vary from ISDN to 100Mbit Ethernets - even within one company. Within this paper, we concentrate on issues that are related to the distribution of manufacturing data.

CommCoop will be implemented as a pilot system, based on already existing systems from different manufacturing companies. The pilot system's functionality will include the central evaluation of manufacturing machines, the efficient control of the manufacturing machines currently at work, error removal and optimization of the machine processes, the transfer of error removal and optimization knowledge between the different machines, and the world-wide access to documentation material like help texts and error descriptions.

3 Taxonomy of Replication

Replication is a process of distributing data to multiple DBs. In the simplest case, this can be achieved by copying schema objects to one or more DBs that make up a distributed DB system. Replication also implies the maintaining of these copies, i.e. keeping them up-to-date.

Replication can improve the performance and protect the availability of applications because data access options can be specified for different complex cases. For example, an application can normally access a DB on a LAN to minimize wide area network traffic and achieve the best performance. Still, the application can continue to function if the local server fails while other servers with replicated data remain accessible. If the connection is re-established, replication conflicts may arise.

Replication can create conflicts when two rows are changed concurrently at different replication sites. Three different types of conflicts that can occur using replication can be distinguished:

Uniqueness conflicts occur when the replication of a row violates entity integrity as for example a primary key. This can happen when two transactions that have been committed at two different sites each insert a row into a respective table replica with the same primary key value. This will cause a uniqueness conflict. New rows are inserted e.g. every time an error occurs at a machine. An easy way to avoid uniqueness conflicts is to append an ID to all primary keys that is different for each participating DB.

Update conflicts arise when two different transactions, that have been committed at different sites, update the same row at nearly the same time. This

occurs if both subsidiary 1 and 2 modify a table, e.g. error messages. There is no way to avoid these conflicts without denying all but one party write-access.

Delete Conflicts occur if at one site a row is deleted that has been updated at another site.

3.1 Related Work

Anderson [1] analyzes how well the *eager* and the *lazy replication* approach perform. Eager replication is the writing of updates to all replicas within a single transaction. Lazy replication, on the other hand, propagates updates to replicas only after the transaction has been committed at the original site. The disadvantage of these well-established replication approaches, however, is that they require locking mechanisms. Locking protocols only work if the DB connection is available all the time.

Daniels [2] investigated on snapshot replication offered by Oracle 7. Even considering advanced features offered by Oracle8i, snapshots are only efficient for certain cases and transfer too much data in all the other cases.

Petersen [7] describes *weakly consistent replication*. The main advantage is that by relaxing data consistency, replication can take place even if DBs are temporarily disconnected. The paper focuses on how replication efficiency potentially decreases when DBs are offline for a long time due to increases in the log files' sizes. Our approach can be considered to be orthogonal to Petersen's in the sense that our taxonomy can also be employed with the weakly consistent replication scheme.

Liu [4] proposes a multiview access protocol for large-scale replication. The protocol organizes DBs into a tree-structured, hierarchical architecture. Similar to lazy replication, the basic concept of the protocol is to replicate data by a set of hierarchically related but independently committed transactions.

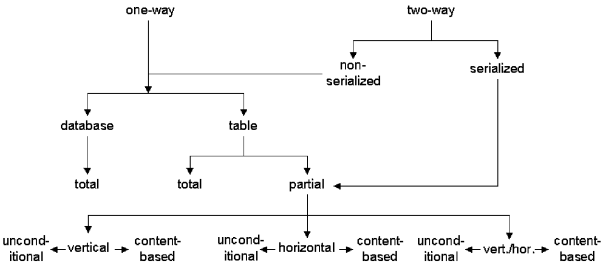


Fig. 2: Taxonomy of Replication Scenarios.

As shown in Fig. 2, we distinguish two basic types of replication: one-way and two-way replication. The following sub-sections go into detail elaborating various kinds of replication.

3.2 One-Way Replication

One-way replication means that data is transferred from the master DB to the slave. This does not imply the slave being read-only. The master may overrule updates made in the slave DB, depending on the replication condition. Two-way replication is more complicated and in certain cases there is no clear solution for update conflicts.

In the following sections, we focus on update conflicts. Delete conflicts can be dealt with in a similar way. The row to be deleted is not deleted but only marked with a special flag. This flag is treated just like another column and updates, which really are 'deletes', are propagated according to replication schema. At predefined intervals all rows marked with the flag are deleted. As previously described, uniqueness conflicts can easily be avoided and are thus of little relevance in the following sections.

Database Replication The whole master DB is replicated to the slave. This implies the slave DB being de facto read-only because every replication destroys all changes made in the slave DB. After every replication, the master DB and the slave DB are identical.

Table Replication This form of replication replicates only certain tables. As these tables are replicated totally, the slave tables are again de facto read-only. The difference to the former replication method is that there may be tables in the master DB and slave DB tables that are not replicated at all.

Vertical Unconditional Table Replication One or more columns are replicated as shown in Figure 3. (Note: In all following figures modifications at the Master are shown in boldface typesetting and modifications at the Slave are underlined. A frame highlights the areas (columns or rows) that are replicated.)

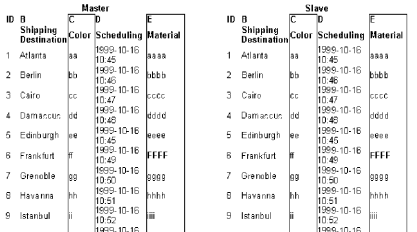


Fig. 3: Vertical Unconditional (Columns C and E) Replication.

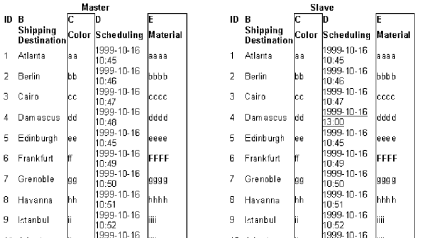


Fig. 4: Vertical Content-based Replication.

The columns C and E are replicated from the master to the slave. After the last replication, the values in bold typeface (cells E6, C12, D15, C16, E16) have been changed at the master, the underlined ones (D4, D15) at the client. Some modifications (all but D15) are propagated to the slave. Not all changes in the master table (D15) will be performed at the slave. Changes in the slave's table (D4, D15) are not overwritten by the master's values. In many cases, the master column that is not propagated to the slave (column D) does not even exist at the master site. For the replication process this does not make a difference.

A real world example is production planning. The master DB is located at the company's headquarters, the slave in a production subsidiary. Column C represents the color of the goods to be manufactured and column D the scheduling information, i.e. when which goods should be produced. The headquarters is not interested in the scheduling, but the production plant obviously has to be informed about changes in the ordered goods' colors. Of course, this example is simplified, because normally this scenario would require a two-way replication, as the headquarters has to be informed about finished goods.

Vertical Content-based Table Replication This replication is only performed if certain conditions that can depend on values in the table or on exterior values hold true. Fig. 4 is very similar to Fig. 3. The important difference is that the modification of C12 is not replicated at the slave due to the condition. The update of this table is only performed for those rows where column B='Pittburgh' or for those where column B='Quito'. The replication also depends on an exterior variable updateEnabled. In this example, there are three different forms of conditions. (1) Conditions referring to content in the columns to be replicated (e.g. if C='QQ'), (2) Conditions referring to content of other columns (e.g. if B='Pittsburgh'), (3) Conditions referring to exterior content (e.g. if updateEnabled).

As in the previous example, the production plan can transmit information on which goods are finished. Updates, i.e. changes of color (C) are not any longer possible for those goods. Goods for Pittsburgh (B='Pittsburgh') or goods with color C='QQ' and all goods with color Q have not yet been manufactured and thus changes are permitted.

Horizontal Unconditional Table Replication Horizontal replication propagates changes in all columns, but only for certain rows to the client. It can also be seen as *content-based vertical replication* where all columns are updated and the condition selects some rows only. An example of horizontal unconditional replication is shown in Fig. 5. The condition for vertical content-based replication would be IF ((ID>11) and (ID<21)) THEN v-replicate B, C, D, E.

We will modify our example a little. The headquarters now also does the scheduling (column D) but it only transmits the data for the current day (ID>11 and ID<21) to the production plant. Replicating older data makes no sense and data for the next day might still change so that there is no point in updating it at the slave before it is really required to do so.

Horizontal Content-based Table Replication If the horizontal replication depends upon a condition, we refer to it as horizontal content-based table replication. An example can be seen in Fig. 6.

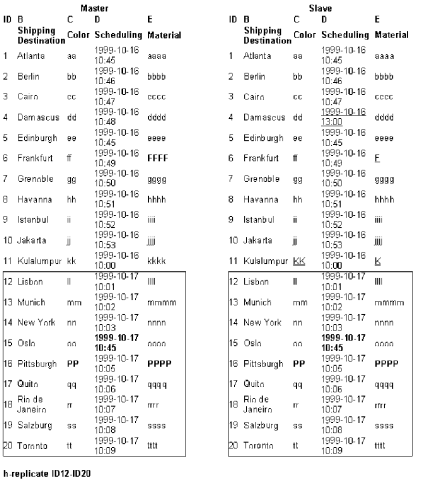


Fig. 5: Horizontal Unconditional Replication.

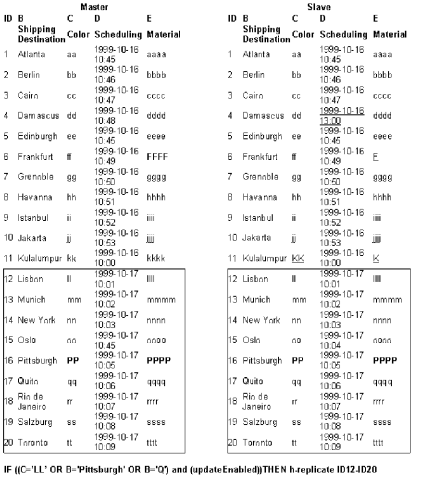


Fig. 6: Horizontal Content-based Replication.

We extend the example given in the last section. Having more than one production plant, in one specific plant only goods for Quito (B='Quito') or Pittsburgh (B='Pittsburgh') and goods that are to be painted in color C='LL' are manufactured. Therefore, only the relevant (content-based) data of the current day (horizontal unconditional ID>11 and ID<21) has to be replicated.

Vertical and Horizontal Unconditional Table Replication Combining vertical and horizontal unconditional replication is very powerful. Only those cells that lie within the specified rows/columns are updated at the slave's DB.

Let us take the example of Horizontal Unconditional Replication. If the production plant does the scheduling (column D) itself, then column D does not have to be replicated. As described in the section dealing with Vertical Unconditional Replication, column D might not even exist at the master's site.

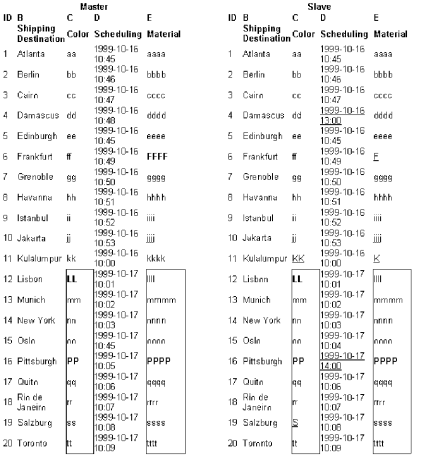
Vertical and Horizontal Content-based Table Replication Adding content-based updates leads to the most complex form of one-way replication. Extending the example of Horizontal Content-based Replication in the same way as done in the last paragraph, gives an example of this last case. The production plant that produces only certain goods receives product information (color) about the products that are to be manufactured every day (current day ID11-ID20). Within this day the scheduling is done in the plant.

3.3 Two-Way Replication

Two-way replication is more complicated than one-way replication. There are two different forms of two-way replication: serialized or non-serialized.

The former is the case if two one-way replication processes are used. For example in the first replication process, DB A is the master, B the slave. After

having finished the first step, the roles are swapped. If cells are updated both in A and B, then the final state depends whether A or B is first master. This situation can easily lead to undesired effects causing the loss of data.



Master

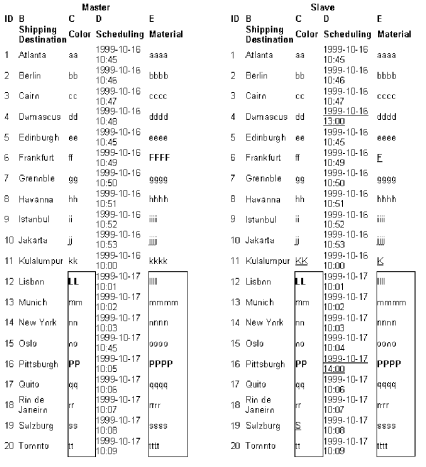
ID	B Shipping Destination	C Color	D Scheduling	E Material
1	Atlanta	aa	1999-10-16 10:45	aaaa
2	Berlin	bb	1999-10-16 10:46	bbbb
3	Cairn	cc	1999-10-16 10:47	cccc
4	Damascus	dd	1999-10-16 10:48	dddd
5	Edinburgh	ee	1999-10-16 10:45	eeee
6	Frankfurt	ff	1999-10-16 10:49	FFFF
7	Grenoble	gg	1999-10-16 10:50	gggg
8	Havanna	hh	1999-10-16 10:51	hhhh
9	Istanbul	ii	1999-10-16 10:52	iiii
10	Jakarta	jj	1999-10-16 10:53	jjjj
11	Kulalumpur	kk	1999-10-16 10:00	kkkk
12	Lisbon	LL	1999-10-17 10:01	llll
13	Munich	mm	1999-10-17 10:02	mmmm
14	New York	nn	1999-10-17 10:03	nnnn
15	Oslo	oo	1999-10-17 10:45	oooo
16	Pittsburgh	PP	1999-10-17 10:05	PPPP
17	Quito	qq	1999-10-17 10:06	qqqq
18	Rio de Janeiro	rr	1999-10-17 10:07	rrrr
19	Salzburg	ss	1999-10-17 10:08	ssss
20	Toronto	tt	1999-10-17 10:09	tttt

Slave

ID	B Shipping Destination	C Color	D Scheduling	E Material
1	Atlanta	aa	1999-10-16 10:45	aaaa
2	Berlin	bb	1999-10-16 10:46	bbbb
3	Cairn	cc	1999-10-16 10:47	cccc
4	Damascus	dd	1999-10-16 13:00	dddd
5	Edinburgh	ee	1999-10-16 10:45	eeee
6	Frankfurt	ff	1999-10-16 10:49	E
7	Grenoble	gg	1999-10-16 10:50	gggg
8	Havanna	hh	1999-10-16 10:51	hhhh
9	Istanbul	ii	1999-10-16 10:52	iiii
10	Jakarta	jj	1999-10-16 10:53	jjjj
11	Kulalumpur	KK	1999-10-16 10:00	K
12	Lisbon	LL	1999-10-17 10:01	llll
13	Munich	mm	1999-10-17 10:02	mmmm
14	New York	nn	1999-10-17 10:03	nnnn
15	Oslo	oo	1999-10-17 10:04	oooo
16	Pittsburgh	PP	1999-10-17 14:00	PPPP
17	Quito	qq	1999-10-17 10:06	qqqq
18	Rio de Janeiro	rr	1999-10-17 10:07	rrrr
19	Salzburg	ss	1999-10-17 10:08	ssss
20	Toronto	tt	1999-10-17 10:09	tttt

vch-replicate C,D : ID11 ID20

Fig. 7: Vertical and Horizontal Unconditional Replication



Master

ID	B Shipping Destination	C Color	D Scheduling	E Material
1	Atlanta	aa	1999-10-16 10:45	aaaa
2	Berlin	bb	1999-10-16 10:46	bbbb
3	Cairn	cc	1999-10-16 10:47	cccc
4	Damascus	dd	1999-10-16 10:48	dddd
5	Edinburgh	ee	1999-10-16 10:45	eeee
6	Frankfurt	ff	1999-10-16 10:49	FFFF
7	Grenoble	gg	1999-10-16 10:50	gggg
8	Havanna	hh	1999-10-16 10:51	hhhh
9	Istanbul	ii	1999-10-16 10:52	iiii
10	Jakarta	jj	1999-10-16 10:53	jjjj
11	Kulalumpur	kk	1999-10-16 10:00	kkkk
12	Lisbon	LL	1999-10-17 10:01	llll
13	Munich	mm	1999-10-17 10:02	mmmm
14	New York	nn	1999-10-17 10:03	nnnn
15	Oslo	oo	1999-10-17 10:45	oooo
16	Pittsburgh	PP	1999-10-17 10:05	PPPP
17	Quito	qq	1999-10-17 10:06	qqqq
18	Rio de Janeiro	rr	1999-10-17 10:07	rrrr
19	Salzburg	ss	1999-10-17 10:08	ssss
20	Toronto	tt	1999-10-17 10:09	tttt

Slave

ID	B Shipping Destination	C Color	D Scheduling	E Material
1	Atlanta	aa	1999-10-16 10:45	aaaa
2	Berlin	bb	1999-10-16 10:46	bbbb
3	Cairn	cc	1999-10-16 10:47	cccc
4	Damascus	dd	1999-10-16 13:00	dddd
5	Edinburgh	ee	1999-10-16 10:45	eeee
6	Frankfurt	ff	1999-10-16 10:49	E
7	Grenoble	gg	1999-10-16 10:50	gggg
8	Havanna	hh	1999-10-16 10:51	hhhh
9	Istanbul	ii	1999-10-16 10:52	iiii
10	Jakarta	jj	1999-10-16 10:53	jjjj
11	Kulalumpur	KK	1999-10-16 10:00	K
12	Lisbon	LL	1999-10-17 10:01	llll
13	Munich	mm	1999-10-17 10:02	mmmm
14	New York	nn	1999-10-17 10:03	nnnn
15	Oslo	oo	1999-10-17 10:04	oooo
16	Pittsburgh	PP	1999-10-17 14:00	PPPP
17	Quito	qq	1999-10-17 10:06	qqqq
18	Rio de Janeiro	rr	1999-10-17 10:07	rrrr
19	Salzburg	ss	1999-10-17 10:08	ssss
20	Toronto	tt	1999-10-17 10:09	tttt

IF ((C='PP' OR B='Quito') and (updateEnabled))THEN vch-replicate C,D : ID11 ID20

Fig. 8: Vertical and Horizontal Content-based Replication

Most replication schemes avoid this problem by globally locking records. The drawbacks are that deadlock probability grows as the forth power of transaction size [1] and that connection between the DBs have to be online.

A non-serialized replication is in fact the 'synchronization' of two DBs simultaneously. Obviously, the serialized and the non-serialized form are identical for tables where the fields copied from DB A to B are read-only in B and vice versa.

In two-way replication, the same cases as in one-way replication can occur, except for serialized DB replication and serialized table replication: (1) Database Replication, (2) Table Replication, (3) Vertical Unconditional, (4) Vert. Content-based, (5) Horizontal Uncond., (6) Horiz. Content-based, (7) Vert. and Horiz. Uncond., and (8) Vert. and Horiz. Content-based.

For most cases, non-serialized two-way replication is needed, we have a closer look at (non-serialized two-way vertical and horizontal unconditional) replication. Fig. 9 (left) shows two DBs before the (non-serialized two-way vertical and horizontal unconditional) replication and Fig. 9 (right) afterwards.

Before the replication, the following modifications were performed: C2, D2, E6, D8 were modified at the master and E7, E8, B15, C15, D17 at the slave.

As only the marked block (D5-E5-D15-E15) is replicated, the changes at the master in E6, E8, D9 are replicated at the slave and vice versa the changes in E9. E8 is a special case. Since changes occurred both at the master and the slave, row 8 was duplicated (row 21) so that the slave's modifications are not lost.

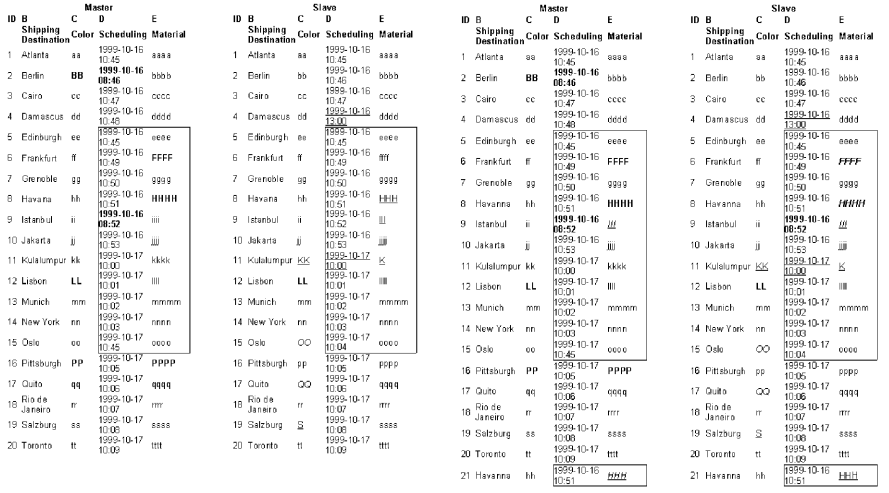


Fig. 9: Two-way Vert. and Horiz. Uncond. Repl. *before* and *after* the replication.

4 Implementation Considering Oracle8i and Java

Being familiar with the various replication concepts, we will now have a look at which existing features can be used to cover the different cases of replication using Oracle 8i.

A replication object is a DB object existing on multiple servers in a distributed DB system. Oracle's replication facility allows to replicate tables and supporting objects such as views, triggers, and indexes.

In a replication environment, Oracle manages replication objects using replication groups. By organizing related DB objects within a replication group, it is easier to administer many objects together. A replication group can exist at multiple replication sites. Replication environments support two basic types of sites: master sites and snapshot sites. A master site maintains a complete copy of all objects in a replication group.

A snapshot site supports read-only and updateable snapshots of the table data at an associated master site. A snapshot site's table snapshots can contain all or a subset of the table data within a replication group. Updateable snapshots can be used for both vertical and horizontal one-way replication. Furthermore, updateable snapshots have the following benefits: (1) allowing users to query and update a local replicated data set even when disconnected from the master site, (2) increased data security achieved by replicating only a subset of the target master table's data, and (3) smaller footprint than multimaster replication.

However, they cannot be used for content-based replication, especially if the condition is not within the DB. For these situations two options are available. Using JDBC guarantees a flexible solution that can be used with all DBMSs

that JDBC supports. The second option is to use DB triggers and stored procedures. The obvious disadvantage is that this approach only works with DBs that support these features. However, our implementation has shown that for high-volume DBs the most flexible, JDBC based replication is too slow — no matter which Oracle JDBC driver we use. Using DB triggers a noticeable gain in speed is achievable as most time-consuming simple conditions that decide whether to overwrite an entry can be evaluated within the DBMS. For the pure JDBC-based approach all data necessary for the evaluation has to be passed through the JDBC interface, which is a considerable bottleneck.

5 Conclusion and Future Work

Our industrial partner AMS Engineering has designed a data model that allows worldwide integration of machine tools. Production data from these machines can be used in the company that owns the machine, at AMS, the manufacturer of the machines, and parts of the non-sensitive data in other companies as well.

The first step has been to consolidate the data model and to create a strategy for centrally configuring and deploying DBs using replication. As many different forms of replication requirements exist, we had to find a sound way of implementation for every single case.

The next step is to integrate the replication concepts into a database agent environment that we currently develop. Unlike Java agents using JDBC, this environment can be used to distribute all DB schema changes necessary for our replication schema.

References

1. Anderson T., Breitbart Y., Korth H.F., Wool A.: Replication, Consistency, and Practicality: Are These Mutually Exclusive? Proc. of the ACM SIGMOD International Conference on Management of Data. (1998) 484–495
2. Daniels D. et al.: Oracle's Symmetric Replication Technology and Implications for Application Design. Proc. of the ACM SIGMOD International Conference on Management of Data. (1994) 467
3. Hellerstein J.M., Haas P.J., Wang H.J.: Online Aggregation. Proc. ACM SIGMOD International Conference on Management of Data. (1997) 171–182
4. Liu X., Helal A., Du W.: Multiview Access Protocols for Large-Scale Replication. ACM Transactions on Database Systems. 23 (2) (1998). 158–198
5. Kemper A., Eickler A.: Datenbanksysteme. R. Oldenbourg. (1997)
6. Oracle 8i Online Documentation. (1998)
7. Petersen K. at al.: Flexible Update Propagation for Weakly Consistent Replication. Proc. ACM Symposium on Operating System Principles. (1997) 288–301
8. Sheth A.P., Larson J.A. : Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys. 22 (3) (1990) 183–236
9. Stonebraker M., Woodfill J., Ranstrom J., Kalash J., Arnold K., Andersen E.: Performance Analysis of Distributed Database Systems. Third Symposium on Reliability in Distributed Software and Database Systems. (1983) 135–138

A Change Impact Analysis Approach For CORBA-Based Federated Databases

L. Deruelle[1], M. Bouneffa[1], N. Melab[1], H. Basson[1],
G. Goncalves[2], and J.C. Nicolas[2]

[1] Laboratoire d'Informatique du Littoral,
B.P. 719. 3, rue Louis David,
62228, Calais, Cedex, France
{deruelle, bouneffa, melab, basson}@lil.univ-littoral.fr
[2] Laboratoire en Organisation et Gestion de Production,
Faculté des sciences appliquées,
Technoparc Futura,
62400 Bethune, France
{goncalves,nicolas}@univ-artois.fr

Abstract. *In this paper, we propose an approach and a framework for an a priori change impact analysis of database schemas, in federated environment. The approach is based on a model, that describes program source codes and database schemas as software components linked by meaningfully relationships. This model takes into account the software components for both centralized and CORBA-based federated database applications. We deal with the federated issue, in accordance with the Object Database Management Group specifications. The change impact analysis is done, using a Knowledge Based System, that includes impact propagation rules, in a distributed way. This is achieved by proposing a framework, that implements our model, in order to simulate the evolution of CORBA-based federated database schemas.*

Keywords: Database Schemas Evolution, Federated Databases, CORBA, Change Impact Analysis, Knowledge Based System.

1 Introduction

Nowadays, observing their information systems, enterprises are confronted with two major problems. The first one concerns the multiplicity of languages, database systems, operating systems and software environments used by the enterprise. This leads to complex applications based on the cooperation of different subsystems. The second problem deals with the frontiers of the information system itself. Supply chain management and e-business challenges leads the information systems to deal with suppliers and customers information systems.

Strictly internal information systems are not sufficiently adequate to bring solutions to these problems. So, distributed complex applications strongly emerge.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 949–958, 2000.

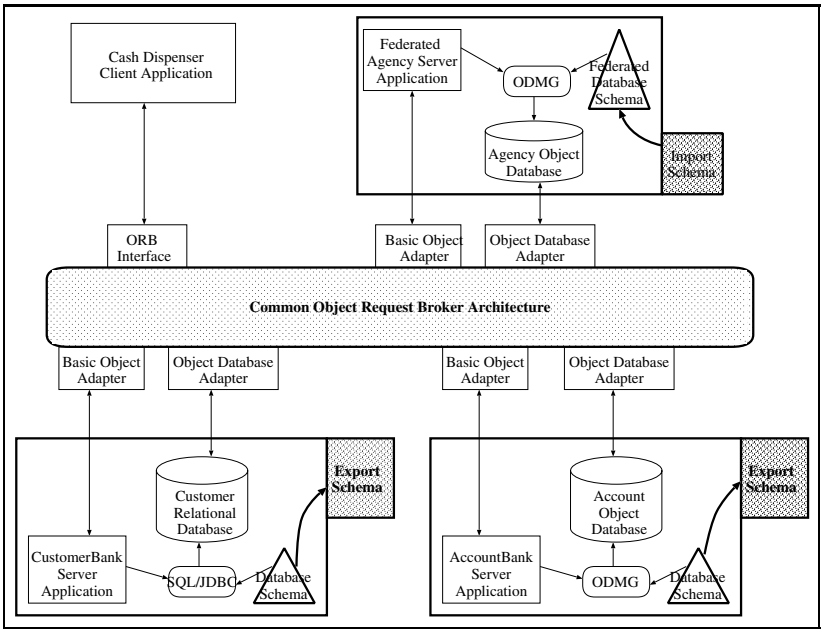


Fig. 1. Distribution of two database schemas based on CORBA middleware

These require complex object modeling, extensibility, integration of large-scale databases and program facilities. Subsequently, distributed system middleware solutions such as CORBA[14], DCOM[4] and Java RMI can be used, as shown on figure 1.

For the federated complex applications, design and implementation seem to be critical tasks. Thus, evolution of such applications is unavoidably crucial and difficult task. Without assisting tools, the software change may cause serious damage to the information system. Without an *a priori* change impact evaluation, it is almost impossible to avoid software quality regression.

In [3], we have proposed a model for the *a priori* change impact analysis for centralized applications, manipulating persistent data. The model represents the database schemas and the program source codes as software component set and their relationships. We have dealt with the *a priori* change impact analysis by using a Knowledge Based System, that includes impact propagation rules, for centralized applications and database schemas evolution. We have extended this model to deal with the federated components and the federated relationships. With this extension, we can analyze the change impact on cooperative databases, in centralized environment. This assumes that both database schemas and program source codes are located on the same site. Nevertheless, this extension needs the programmers to define the federation architecture, following our model. This leads to rewrite the federation architecture[3]. In this paper, we refine the relationships in order to deal with evolution of CORBA-based federated database

systems, using the standard services of the Object Management Group[14], and without to rewrite the federation architecture.

In comparison with other models [1][12], the major advantage of our approach concerns the deal with multilanguage programs source codes and the federated databases, in both centralized and distributed environment. Our approach consider both the program source codes and database schemas, in an homogeneous representation. This allows to perform change propagation of database schemas to all components dealing with it. Most of the models proposed in the literature do not integrate multilanguage programs, manipulating data stored in federated databases. This leads to consider the evolution of programs source codes, or database schemas, in separate manner. We propose a framework that implements our model and performs the change impact propagation on the different components of the application, especially the federated databases components. The software evolution area has led to a lot of research and industry work. The most significant ones concern the evolution of source codes [12] or database schemas[5] in a separate manner. To deal with adaptating existing programs to schema changes, some authors introduce a kind of *ad hoc* polymorphism[11]. However, they always consider an application as a collection of programs written by means of the same language that supports such a polymorphism. Otherwise, the evolution of distributed and federated databases have always been considered as the conceptual level [13]. In this paper, the challenge is to propagate the change of any database schemas to all the distributed components dealing with it.

The paper is organized as follow: the section 2 presents our generic model, called Software Components Structural Model, and the associated knowledge based system. The section 3 shows the refinement of the model for the change impact analysis of CORBA-based federated database systems. The section 4 concerns the global architecture of our framework, that implements the extended model. In the section 5, we illustrate the global mechanism with an example. Finally, section 6 gives a conclusion and some perspectives of our works.

2 SCSM a Generic Model for the Change Impact Analysis

2.1 The Basic Elements of the Model

The Software Components Structural Model (SCSM) is the central element of our *a priori* change impact analysis approach. It represents the various software components and their relationships. The software components are classified following a granularity criterion :

1. *the coarse grained components* that may be source code files, databases schemas, libraries, and packages,
2. *the middle grained components* that may be persistent or transient classes, interfaces, data members, methods, defined in programs or database schemas,
3. *the fine grained components* like statements, queries, and individual symbols.

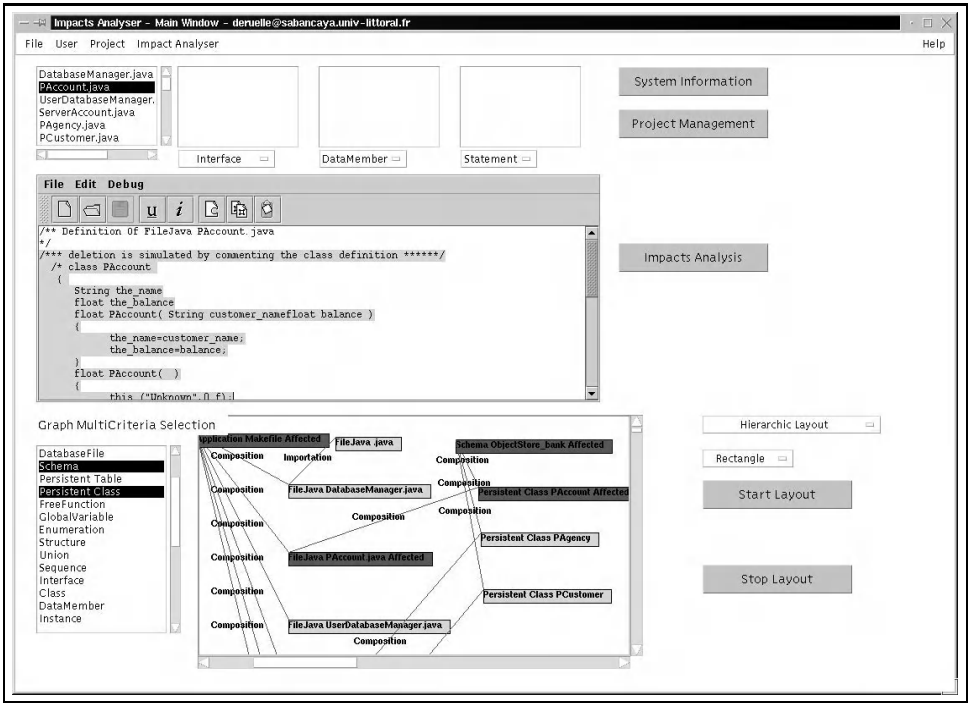


Fig. 2. Example of a change impact analysis result : The affected nodes are labeled

The components are linked together by the use of specific relationships : *the Importation Relationship* between files, schemas or packages, *the Inheritance Relationship* that exists between the classes of programs or database schemas, *the Call Relationship* between a function or a method.

A more detailed description of the model can be found in [9].

The set of software components and their relationships are formalized by the way of multigraph[3]. The nodes are the software components and the edges the associated relationships. The Figure 2 gives an example of such multigraph.

2.2 The Knowledge Based System

In order to perform a change impact propagation, we define a knowledge based system. This includes facts as software components, and impact propagation rules based on relationships. The rules represent the knowledge of the evolution experts. The rules are defined in accordance with a change typology [3]. When a user changes a software component, some rules are fired depending of the kind of the change and depending of the existing relationships in the multigraph. This mechanism allows to deduce the software components affected by the programmer change simulation.

For example, the figure 2 shows the change impact propagation result on the multigraph. The multigraph represents a database schema and source codes of

a bank application, in a centralized environment. In this example, We assume that both schema definition and program source codes are located on the same site. The database schema describes three Persistent classes, called *PAccount*, *PCustomer* and *PAgency*. This result is based on the deletion of the persistent class PAccount. This is performed by commenting the persistent class definition, in the code source editor. This change allows the knowledge based system to propagate the impact directly on the multigraph nodes. The affected nodes are marked, and their label contains the *affected* keyword.

In the following section, we present the previous extension of the Software Components Structural Model, regarding the federated databases.

2.3 Previous Extension Of The Software Components Structural Model

In [3], we have proposed a basic approach for the *a priori* change impact analysis, in a federated databases environment. This needed the definition of new relationships between software components (the federated relationships) and, the enrichment of the KBS with specific federated propagation rules. The federated propagation rules take into account the relationships between federated components, in order to propagate the impacts at the federated level[3]. So this solution provides a mean to estimate the change impact through federated database systems. Nevertheless, the programmers need to describe the federated relationships: how software components are linked in the federation. In other words, they need to make a SCSM representation of the pre-existing federation information. This leads to rewrite the federation, instead of using the *import/export schemas* [13]. We propose to exploit the informations, provided by the *import/export schemas*, in order to lean on the federation architecture. This needs to distribute the multigraphs, in accordance with the federation architecture. Furthermore, the previous extension does not deal completely with distributed applications, since it is fit for use over federation of local databases.

In order to face the strong emergence of distributed applications based on federated databases, we propose in the next parts to refine the software components and the relationships. The goal of this is to allow impact propagation through CORBA-based federated databases, using the federation architecture.

3 The Software Components Structural Model for the Corba-based Federated Databases

This extension leads to represent the software components, based on the Object Database Management Group [2] point of view. The new software components are classified following the granularity criterion :

1. *The ODL file* that defines the database schema, in accordance with the Object Definition Language. The ODL defines a particular syntax nearby of the C++ language and introduces the relationship concept[2]. The ODL is a superset of the previous IDL [14].

2. *The Object Database Adapter*, that allows the database connection with the ORB[2], providing to access to a huge volume of objects.
3. *The Persistent Class*, that defines the state and the behavior of the objects, stored in the database.
4. *The Collection* that allows to represent the result set provided by the object query.
5. *The Object Query* provides querying of object database systems, including high-level primitives for object sets and structures.

These new components are linked, using the following relationships :

1. *The Projection Relationship*, that exists between the ODL specifications components and their representations on the implementation languages (C++, Java, and so on).
2. *The Manipulate Relationship* between the Object Query components and the Collections.

This extension is implemented by a distributed framework, providing change impact propagation. Doing this, we deal with the federated and heterogeneous database system, since our previous model yet includes the relational database representation.

We present, hereafter, the distributed framework for the change impact analysis, and we illustrate the change propagation on the CORBA-based bank application.

4 The Change Impact Analyzer Framework Architecture

In complex distributed applications, the program source codes and the databases schemas are distributed over many computers and networks. For the change impact analysis, the software components are distributed, in the same way. In order to take into account the federated relationships between software components, especially database schemas, we need to distribute our multigraphs, following the federation architecture. The *import/export schemas* describe the federated components architecture [13]. These schemas are usually stored in repository. In our change impact analyzer framework, we can access this repository, or allows the programmers to describe theses schemas as XML files[10]. These allow to collect information about the federated relationships, in order to allow the multigraphs distribution. The distributed multigraphs are connected to others, by the way of distributed agents. The distributed agent allows to propagate the change impact to other distributed agents, when the knowledge based system has marked the local multigraph nodes. The distributed agents insert the new received facts, inside the knowledge based system, to perform the change impact analysis process at the federated level.

The model have been implemented by a framework, called *Integrated Framework for Software Evolution and Maintenance*. This provides change impact propagation for both centralized and distributed software. The figure 3 shows

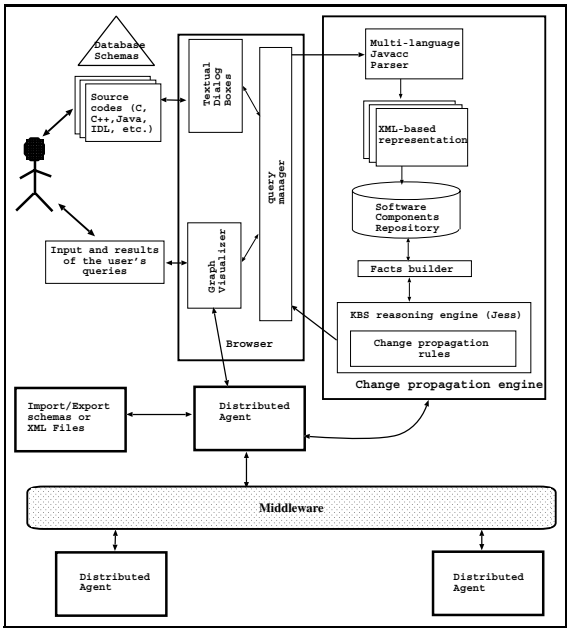


Fig. 3. The Distributed Framework Architecture for Change Impact Propagation

the framework architecture, for distributed change propagation. It provides incremental and multilanguage source code files and distributed database schema files parsing to extract structural informations, using *Javacc parser*. The structural information is represented by XML files, that are stored in a distributed component repository. The component repository is implemented with *ObjectStore Database*, allowing to stores complex object graphs. The *query manager* allows the programmer to simulate its changes. The change impact propagation is computed by the *Change Propagation Engine*, using the Expert System, called *Jess*[3]. This propagates locally the change impact, using propagation rules, and marks the multigraph nodes, with the *affected* label. In a distributed environment, the *Jess* Expert System propagates the change impact, using the *distributed agent*. This broadcasts the affected components set to other distributed agents. These receivers insert the new facts in their local *Change Propagation Engine*, in order to perform change impact analysis, at a distributed level. The receivers distributed agents broadcast the result of the change impact analysis, in order to inform the programmer of the change effects, at the distributed level.

We present, hereafter, an example of change impact analysis, performed on a CORBA-based bank application.

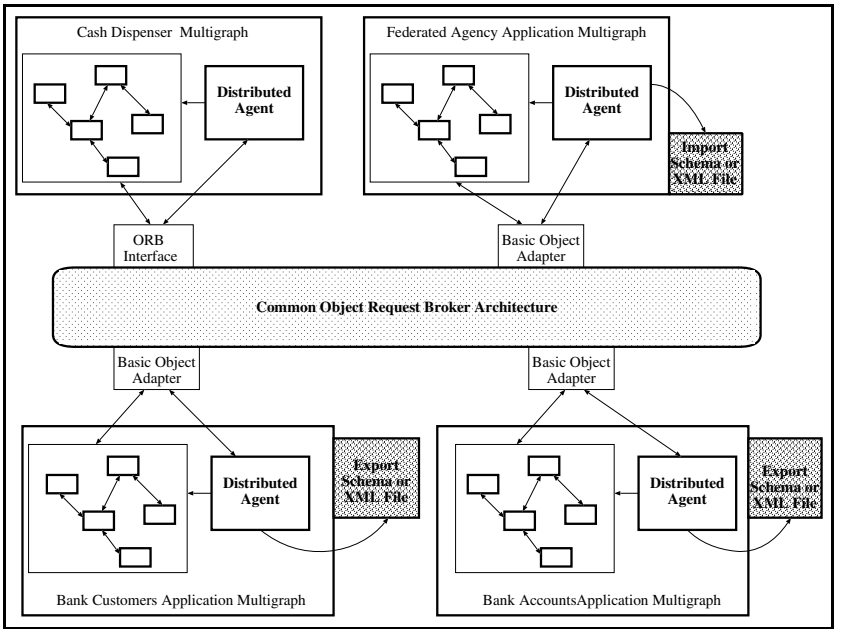


Fig. 4. Multigraph representation of the federated components

5 Application : Evolution of CORBA-based bank Federated Databases

We propose an example of the change impact analysis for the bank application based on CORBA. The figure 1 shows a global view of the bank application. This is composed of three distributed and heterogeneous database schemas :

- *The Agency schema*, is an *ObjectStore* ODMG-compliant database, that stores all transactions related to the customers and their accounts. The Agency database system is running on a Pentium II 350MHz with a Linux Operating System.
- *The Customers schema*, is an *Oracle 8i* relational-object database, that stores the bank customers, related to an agency. The Oracle database is running on a Pentium II 450MHz with a Linux operating system.
- *The Accounts schema*, is an *ObjectStore* ODMG-compliant database, that stores all bank accounts related to an agency. This database is running on a Silicon Graphics workstation with IRIX5.3 operating system.

These database schemas are connected by the Object Request Broker, called ORBACUS[7].

The *Agency Application* manages the customers and the bank accounts, in a distributed manner. The *Agency Application* provides customers operations, like customers registration, and accounts operations like debit and credit. The *Cash*

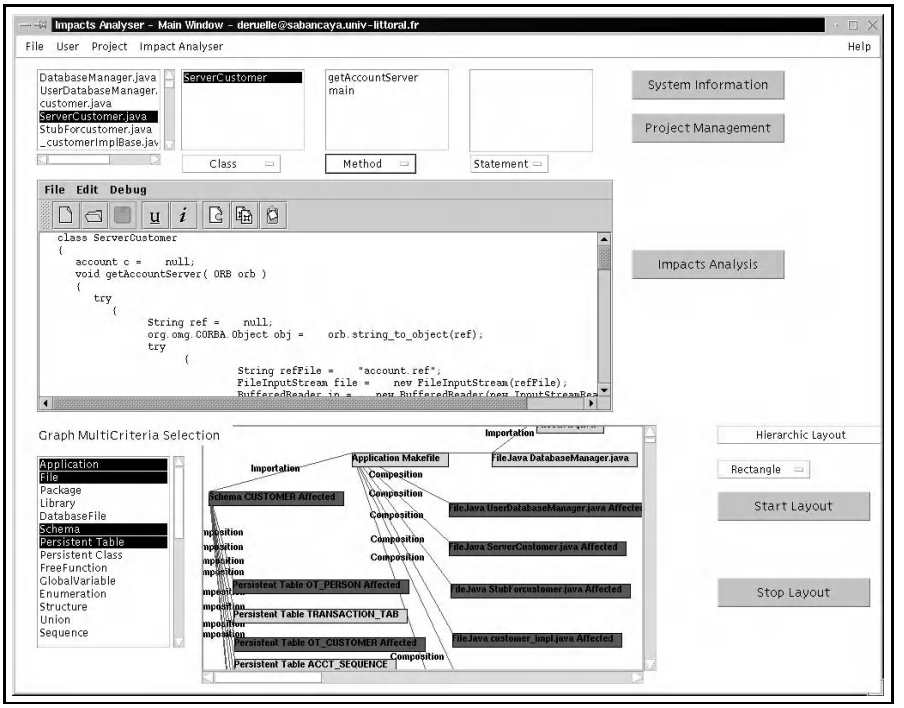


Fig. 5. Result of the change impact propagation at the federated level

Dispenser allows to query the Agency Application to perform debit or credit operations. The figure 4 shows the distributed multigraphs of the CORBA-based bank application. These are interconnected, using distributed agents based on CORBA. The distributed agents lean on the *import/export schemas* or the *XML files*, that describe the federation architecture.

In this example, we propose to perform a change, related to the Account schema. The considered change is the Persistent Class *PAccount* deletion. The Change Propagation Engine computes the change impact propagation locally. The distributed agent propagates the change impact to others, using the CORBA middleware. The receivers distributed agents insert the affected components in the Knowledge Based System, in order to perform change impact analysis. The figure 5 shows the change propagation on the Customer Relational database schema, using the Agency database schema at the federated level.

6 Conclusion

We have proposed and implemented an approach for the *a priori* software change impact analysis. This approach is based on a model, representing the multilanguage programs source codes and the federated heterogeneous database schemas. The model has been formalized with a multigraph, in which the nodes are the

software components and the edges their relationships. The model has been refined to represent the CORBA-based federated databases, in accordance with the Object Database Management Group. The proposed approach lean on the federation architecture, following the *import/export schemas*, according to an approach proposed by Sheth and Larson[13]. Thus, the multigraphs have been distributed, following the federation architecture. This allows to perform the change impact propagation through the federation.

The framework that implements our approach is based on the use of a knowledge based system (KBS), that makes it more flexible. In order to perform the change impact analysis. Subsequently, the rules set of the KBS is continuously enriched in order to achieve more exhaustiveness and to consider several aspects of software evolution like analyzing the change impact on the software performances [8].

References

1. D.C. Atkinson and W.G. Griswold. The Design of Whole-Program Analysis Tools. In IEEE Computer Society, editor, *The proc. of the 18th International Conference on Software Engineering, Berlin*, March 1999.
2. Rick G. G. Cattel. *ODMG-93 Object-Oriented Databases Standard.* International Thomson Publishing, 1995.
3. L. Deruelle, M. Bouneffa, J.C. Nicolas, and G. Goncalves. Local and Federated Database Schemas Evolution: An Impact propagation Model. In Lecture Notes in Computer Science, editor, *The proc. of the Database and Expert Systems Applications*, 1999.
4. G. Eddon and H. Eddon. *Inside Distributed COM.* Microsoft Press, 1999.
5. Fabrizio Ferrandina, Thorsten Meyer, and Roberto Zicari. Schema and database evolution in the o2 object database system. In *Proceedings of the 21th International Conference on Very Large Databases, Zurich, Switzerland (VLDB '95)*, September 1995.
6. http://www.ooc.com/. *ORBACUS : user guide*, December 1999.
7. N. Melab, H. Basson, M. Bouneffa, and L. Deruelle. Performance of Object-oriented Code: Profiling and Instrumentation. *Proc. of the IEEE Intl. Conf. on Software Maintenance (ICSM'99)*, Oxford, UK., Aug. 30 - Sep. 3 1999.
8. N. Melab, L. Deruelle, M. Bouneffa, and H. Basson. Change propagation in multilanguage distributed software. *Proc. of ISCA-PDCS'2000, Las Vegas, Nevada, USA*, Aug.08–10. 2000.
9. A. Michard. *XML langage et applications.* Editions Eyrolles, ISBN 2-212-09052-8, 1999.
10. S.L. Osborn. The role of polymorphism in schema evolution in an object-oriented database. *IEEE Transactions on Knowledge and Data Engineering*, 1(3):310–317, 1989.
11. V. Rajlich. A Model for Change Propagation Based on Graph Rewriting. *Proc. of IEEE-ICSM'97, Bari, Italy*, pages 84–91, Oct. 1–3 1997.
12. A.P. Sheth and J.A. Larson. Federated databases systems for managing distributed, heterogenous, and autonomous databases. *ACM Computer Surveys*, pages 183–236, September 1990.
13. R. Zahavi and T. Mowbray. *The Essential CORBA-Systems Integration Using Distributed Objects.* John Wiley and Sons, Inc, 1996.

A qualitative formalization of built environments*

Thomas Bittner

Centre de recherche en geomatique,
Laval University, Quebec, Canada
Thomas.Bittner@scg.ulaval.ca

Abstract. In this paper I argue that a qualitative formalization of built environments needs to take into account: (1) the ontological distinction between bona-fide and fiat boundaries and objects, (2) the different character of constraints on relations involving these different kinds of boundaries and objects, (3) the distinction between partition forming and non-partition forming objects, and (4) the fundamental organizational structure of regional partitions. I discuss the notion of boundary sensitive rough location and show that a formalization based on this notion takes all these points into account.

1 Introduction

Imagine a computer program (I) that generates configurations of lines in the Euclidean plane that look like plans of built environments [Lyn60]. Examples of built environments are shopping malls, airports, or parking lots (See, for example, the left part of Fig. 1.). Program (I) generates configurations of lines of different style and width. Suppose our task is to design another computer program (II), which checks (1) whether or not a plan generated by (I) can possibly be a plan of a built environment and, in case the plan represents a built environment, (2) whether or not this environment can be navigated easily by human beings.

Human navigation and wayfinding in general and in built environments in particular has been studied extensively in the past in architectural design, e.g., [GLM83], in Artificial Intelligence, e.g., [Kui78] and in Cognitive Science, e.g., [SW75]. Notice that all those people deal with navigation in real, physically existing environments. The question I am addressing is different. The built environments this paper is dealing with do not (physically) exist yet. Consequently, the approach I am proposing cannot rely on observations in reality. The only 'physical' thing we have is a plan, P, generated by program (I). Only if program (II) decides (a) that P represents a built environment and (b) it is easy to navigate then the environment is being built according to P. Whatever it means to be a built environment and whatever it means to be easy to navigate, it must be definable in the language of P and it must be decidable given P. Consequently, task (1) and (2) rest upon the same formal foundations and are, in this respect, closely related to each other. It is the aim of this paper to investigate those formal foundations.

Program (I) produces a quantitative representation of built environments based on computational geometry (e.g., [Rou94]). The analysis of plans representing built environments, performed by program (II) will focus on qualitative aspects [CBGG97],

* The financial support from the Canadian GEOID network is gratefully acknowledged.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 959–969, 2000.

i.e., different kinds of things, qualitative relations between lines [All83] and qualitative relations between regions [CBGG97]. At the formal level I will use a language based on the qualitative notion of boundary sensitive rough location [BS98] to describe built environments. I am going to show that this notion provides the basis for the formal description of built environments and for the evaluation of the complexity of navigation.

This paper is structured as follows. I start with an informal analysis of the ontological makeup of built environments. In Section 3 I shortly review the notion of rough approximations regions and rough location of spatial objects. These notions provide the basis for the formalization. In Section 4 I give a formalization of built environments. The conclusions are given in Section 5.

2 Built environments

In this paper I use parking lots as a running example for built environments. The parking lot domain is relatively simple but its structure is rich enough to study the *ontological* makeup of built environments, which is critical for a *qualitative* formalization. An example of a parking lot in a bird's eye view is given in Fig. 1.

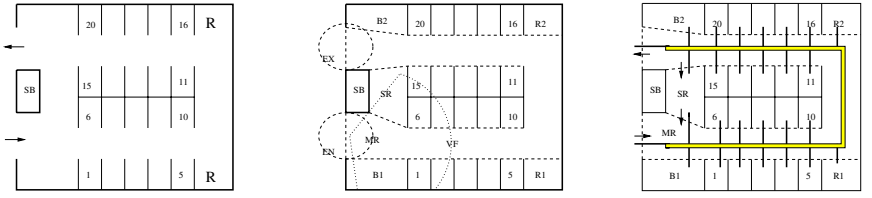


Fig. 1. An empty parking lot (left). The parking lot with invisible fiat boundaries marked (middle). The path system of the parking lot (right).

2.1 Boundaries

Following [Smi95] I distinguish bona-fide and fiat boundaries. Bona fide boundaries are boundaries *in the things themselves*. Bona fide boundaries exist independently of all human cognitive acts. They are a matter of qualitative differentiations or discontinuities of the underlying reality. Examples are surfaces of extended objects like cars, walls, the floor of a parking lot. Bona-fide boundaries are marked by bold solid lines in the left and middle parts of Fig. 1.

[Smi95] describes fiat boundaries as boundaries which exist only in virtue of different sorts of demarcation effected cognitively by human beings. Such boundaries may lie skew to boundaries of *bona-fide* sort as in the case of the boundaries of a parking spot in the center of a parking lot, e.g. spots 6-15 in Fig. 1. They may also, however as in the case of a parking spot at the outer wall of the parking lot, involve a combination

of fiat and *bona-fide* portions such as a wall at its back side, e.g., spots 1-5 and 16-20 in Fig. 1.

Fiat boundaries like the front boundaries of parking spots are not directly observable in reality but do nevertheless exist. Since fiat boundaries are not perceivable by the senses they need to be made visible or the environment must force all people to perceive them in places the designer wanted them to be. In order to make parking spots perceivable by other people, the back, left, and right boundaries are marked by (usually white) paint (the thin solid lines in Fig. 1). The front boundary of the parking spot is not marked but every human being knows that it is located on the straight line connecting the ends of the left and right boundaries. Non marked and hence invisible fiat boundaries are marked by dashed lines in the middle part of Fig. 1. Plans of built environments to be analyzed by programs rather by human beings must contain *all* boundaries even if they are not directly observable in reality. Consequently, plans of built environments generated by program (I) and analyzed by (II) must look like the middle part of Fig. 1 rather than like the left part.

Consider the parking lot domain. Marked fiat boundaries afford people (in cars) not to cross despite the fact that there is no physical barrier. Non-marked boundaries afford crossing, e.g., the non-marked boundary of an empty parking spot 'invites' you to cross this boundary and park your car at this spot (if there is no other car parked yet). In the design of built environments fiat boundaries and their barrier properties play an important role. They provide an important organizational structure. In this paper I distinguish barrier and non-barrier boundaries. Bona-fide boundaries are barrier boundaries. Fiat boundaries may be of barrier (from one side or both sides) or non-barrier sort.

2.2 Spatial objects forming built environments

The classification of boundaries generalizes to a classification of objects. Bona-fide objects have a single topologically closed bona-fide boundary (e.g., the building SB in Fig. 1). Fiat objects have fiat boundary parts (e.g., parking spot 1). Built environments are populated by bona-fide as well as by fiat objects. There are three basic classes of axioms governing the spatial objects in built environments: $(O1)$ axioms governing spatial objects of the *same* ontological kind; $(O2)$ axioms governing spatial objects of the *different* ontological kind; $(O3)$ domain specific axioms characterizing built environments like parking lots, airports, shopping malls, or city centers. In the remainder I call the axioms $O1 - O3$ ontological axioms or ontological constraints on relations that can hold between objects in built environments.

The main axiom in group $O1$ is that spatial objects of the *same* ontological kind cannot overlap [CV95]. For example, bona-fide objects like cars and walls cannot overlap. Fiat objects of the same kind like parking spots cannot overlap either. This axiom needs refinement regarding overlap of boundary parts: Boundary parts of fiat objects of the same kind can be co-located, e.g., co-located boundary parts of neighboring parking spots. Boundary parts of bona-fide objects cannot [SV99].

The main axiom in group $O2$ is that spatial objects of *different* ontological kind can overlap [CV95]. This is not a constraint. It rather says that in general there are no ontological objections against objects of different kinds to overlap. However, there

are additional constraints on relations among partition forming objects (to be discussed below).

There are further *domain specific* axioms, $O3$, constraining relations that can hold between objects of ontological different kind in specific built environments. Consider the parking lot domain. There are cars and parking spots. Parking spots are such that cars can be parked in them. Parking lots are also formed by objects like blocked areas and reserved parking spots (e.g., B_1 and R_1 in Fig. 1). Examples for domain specific constraints on relations between objects in parking lots are: $(S1)$ Cars are supposed to keep blocked areas clear; $(S2)$ Regular cars should not be parked in reserved parking spots; $(S3)$ Cars should be parked within parking spots.

Regarding domain specific constraints in $O3$ it is important to notice that constraints involving objects of ontological different kind are weaker than the constraints between bona-fide objects, constraints between fiat objects of the same kind. The laws of logic prohibit objects of ontological same kind to overlap. Laws of physics prevent bona-fide objects from sharing boundary parts. Constraints involving fiat objects of ontological different kind are based on social rules and agreement and may be violated in certain situations. For example, you can die if you try to drive through a wall. You only get charged when you are parking on a reserved parking spot. The different character of constraints will play an important role in the formalization in Section 4.

Besides the (fundamental) distinction between bona-fide and fiat objects I distinguish two kinds of objects in built environments: (regional) partition forming objects and non-partition forming objects. A regional partition is a set of regions, which members intersect only at their boundaries $(P1)$ and, as a whole, sum up the whole space $(P2)$. Partition forming objects are spatial objects, which form a regional partition of the underlying space.

Consider the parking lot domain. The partition forming objects form a regional partition of the three dimensional parking lot. Each of those objects carves out three dimensional regions of the parking lot but there is no 'no man's land' and no 'double occupation'. Partition forming objects are, for example, parking spots, traffic lanes, sidewalks, blocked areas keeping fire exits clear, walls, pillars, and others more. Partition forming objects may be of bona-fide (pillars or walls) or of fiat sort (parking spots).

Non-partition forming objects overlap partition forming objects of ontological different kind. Non-partition forming objects may be of bona-fide or fiat sort. Consider the parking lot domain. Examples for non-partition forming bona-fide objects are cars and people. Examples for non-partition forming fiat objects are smoking areas in public places, the visual field (VF) in a given location or 'the entrance area', EN, or the 'exit area', EX of a parking lot (See middle part of Fig. 1).

The formalization presented in this paper will be dealing with 2-dimensional objects in 2-dimensional space, i.e., with orthogonal projections of three dimensional objects onto the ground. Consequently partition forming objects form regional partitions of the plane. In this context I am assuming that the projected objects 'inherit' the ontologically significant properties of their originals as well as the barrier and non-barrier properties the boundaries of their originals.

2.3 Movement

An important aspect of the distinction between partition forming and non-partition forming objects is that the partition structure is static and that non-partition forming objects can move (like cars), or shrink and grow (like the visual field). Objects move along paths. A path is a sequence of locations occupied at consecutive moments of time, which corresponds to continuous movement.

Consider the parking lot domain. It is the purpose of a parking lot to let cars park within parking spots. In order to fulfill this purpose, it must be possible to move a car from the entrance to a free parking spot. That is: (i) There must *exist* a path of movement to a free parking spot without violation of $O1 - O3$. This will be called the moveability axiom, M, of a built environment. (ii) It must be possible for a human agent in a car to *find* this path. Checking the existence of paths is an instance of problem (1). Deciding whether or not it is possible, difficult, or easy to find an existing path is an instance of problem (2). In this paper I focus on problem (1). This provides the basis for solving problem (2).

3 Approximating regions and relations between approximations

In this section I shortly discuss the formal notions needed in the remainder of this paper. These notions were originally introduced in [BS98] and [BS00]. For an extended discussion see also [Bit99].

3.1 Approximations

Suppose we have a space R of detailed or precise regions. By imposing a partition, G, on R we can approximate elements of R by elements of ${\Omega_{bs}}^{G\times G}$. That is, we approximate regions in R by functions from $G \times G$ to the set $\Omega_{bs} = \{\mathsf{fo}, \mathsf{fbo}, \mathsf{pbo}, \mathsf{nbo}, \mathsf{no}\}$. The function which assigns to each region $r \in R$ its boundary sensitive approximation is $\alpha : R \rightarrow {\Omega_{bs}}^{G\times G}$. The value of $(\alpha r)(g_1, g_2)$ is fo if r covers all of the cell g_1, it is fbo if r covers all of the boundary segment, (g_1, g_2), shared by the cell g_1 and g_2 and some but not all of the interior of g_1, it is pbo if r covers some but not all of the boundary segment (g_1, g_2) and some but not all of the interior of g_1, it is nbo if r does not intersect with boundary segment (g_1, g_2) and some but not all of the interior of g_1, and it is no if there is no overlap between r and g_1. Consider the visual field, VF, in the parking lot in Fig. 1. The graph of the mapping $\alpha(VF)$ contains the following tuples:

(g_i, g_j)	(B_1, MR)	(B_1, PS_1)	(B_1, W)	(PS_1, B_1)	...	(PS_1, W)	...
ω	pbo	fbo	pbo	fo	...	fo	...

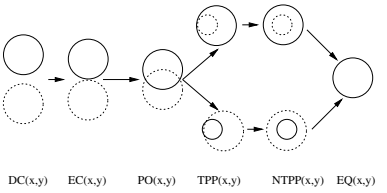
Each approximate region $X \in {\Omega_{bs}}^{G\times G}$ stands for a set of precise regions, i.e., all those precise regions having the approximation X. This set, $[\![X]\!]$, provides a semantics for approximate regions: $[\![X]\!] = \{r \in R \mid \alpha r = X\}$. In the remainder I use the notion *boundary sensitive rough location*, $(\mathrm{loc}\ o) = (\alpha_5 \circ r)o$ in order to refer to the approximation of the (exact) region[1], $r(o)$, of the object o with respect to an underlying partition G.

[1] The region of space it exactly occupies.

3.2 Relations between approximations

In the domain of regions we distinguish a set of 8 well known binary topological relation between spatial regions, the RCC8 [CBGG97] relations.

We distinguish the relations $DC(x,y)$ (disconnected), $EC(x,y)$ (externally connected), $PO(x,y)$ (partial overlap), $TPP(x,y) = TPPi(y,x)$ (tangential proper part), $NTPP(x,y) = NTPPi(y,x)$ (non-tangential proper part), and $EQ(x,y)$.



In the remainder I use the notion $RCC8$ in order to refer to this set of relations. The elements of the set are jointly exhaustive and pairwise disjoint [CBGG97] and form a lattice with respect an ordering relation $\leq$ [BS00]. Possible geometric interpretations and the ordering relation ($R_1 \leq R_2$ is indicated by an arrow from R_1 to R_2) are shown in the figure above.

Let X and Y be boundary sensitive approximations. [BS00] showed that there exists a mapping

$$\Psi \ : \ {\Omega_{bs}}^{G\times G} \times \ {\Omega_{bs}}^{G\times G} \quad \rightarrow \quad RCC8 \times RCC8$$

such that $\Psi(X,Y) = (R^8_{min}, R^8_{max})$ if and only if (1) $R^8_{min}(x,y)$ is the least RCC8 relation that can hold between $x \in [\![X]\!]$ and $y \in [\![Y]\!]$, (2) $R^8_{max}(x,y)$ is the greatest RCC8 relation that can hold for x and y as above, and (3) for each R with $R^8_{min} \leq R \leq R^8_{max}$ there are $x \in [\![X]\!]$ and $y \in [\![Y]\!]$ such that $R(x,y)$, where $\leq$ is the ordering shown in the figure above. For details see [BS00]. In the remainder I use the notions $R^8_{min}(X,Y)$ in order to refer to the least relation and $R^8_{max}(X,Y)$ in order to refer to the greatest relation that can hold between $x \in [\![X]\!]$ and $y \in [\![Y]\!]$.

4 Formalizing built environments

Formally, built environments have three major components: The layout of the built environment, which is formed by the partition forming objects; A system of paths along which non-partition forming objects can move within the layout of the built environment; A set of possible situations, where a situation in a built environment is the layout of the environment and a set of non-partition forming objects populating it in a given moment of time.

Situations need to obey the ontological axioms, $O1 - O3$ and the partition axioms, $P1 - P2$. Furthermore they need to be such that the non-partition forming objects populating the environment could possibly have been moved into the location they are in this situation (axiom M). In this section I give axioms for situations in built environments. These axioms take into account: (1) The distinction between bona-fide and fiat objects; (2) The distinction between partition forming and non-partition forming objects; (3) The different strength of constraints on relations involving bona-fide and fiat objects. Formally, the axioms characterizing built environments are given in terms of boundary sensitive rough location.

4.1 Formalizing ontological constraints

Bona-fide objects do not overlap and do not have co-located boundary parts. Let o_1 and o_2 be bona-fide objects, i.e., $o_1, o_2 \in BF$[2]. In terms of rough location we define: $F1(o_1, o_2) \equiv R^8_{min}((\text{loc}_{bs}\ o_1), (\text{loc}_{bs}\ o_2)) = DC$ and postulate $\forall o_1, o_2 \in BF : o_1 \neq o_2 \Rightarrow F1(o_1, o_2)$. Bona-fide objects can be located in a built environment such that the minimal relation between their exact regions, which is consistent with their rough location $(\text{loc}_{bs}\ o_1)$ and $(\text{loc}_{bs}\ o_2)$, is disconnected, i.e., $DC(r(o_1), r(o_2))$. There cannot exist an environment that forces bona-fide objects to be connected[3].

Two bona-fide objects cannot be connected even if they share the same rough location. In terms of rough location it is impossible to postulate that bona-fide objects cannot be connected. Consider Fig. 1 and imagine two cars on the main road. Both share the same rough location and we have $R^8_{max} = EQ$. In terms of rough location we cannot exclude the possibility for the cars to be connected. Notice the important point: In terms of rough location we specify what an environment *cannot do* to bona-fide objects populating or forming it - it cannot make them being connected. The objects themselves are governed by the underlying theory of objects.

Fiat objects of the same kind do not overlap but may have co-located boundary parts. Let o_1 and o_2 be fiat objects of kind ϕ, i.e., $o_1, o_2 \in F^\phi$[4]. In terms of rough location we define: $F2(o_1, o_2) \equiv R^8_{min}((\text{loc}_{bs}\ o_1), (\text{loc}_{bs}\ o_2)) \leq EC$ and postulate $\forall o_1, o_2 \in F^\phi : o_1 \neq o_2 \Rightarrow F2(o_1, o_2)$. There cannot exist a built environment that forces fiat objects of the same kind to overlap. In terms of rough location it is impossible to postulate that fiat objects of the same kind cannot overlap. This is the business of the theory of objects.

Partition forming objects. Let o_1 and o_2 be bona-fide partition forming objects. In terms of boundary-sensitive rough location we define: $F3(o_1, o_2) \equiv R^8_{max}((\text{loc}_{bs}\ o_1), (\text{loc}_{bs}\ o_2)) = DC$ and postulate $\forall o_1, o_2 \in BF : (o_1 \neq o_2 \text{ and } r(o_1), r(o_2) \in G) \Rightarrow F3(o_1, o_2)$. Due to the underlying partition structure we are able to postulate that partition forming bona-fide objects cannot be connected. The largest relation that can hold between two partition forming bona-fide objects is DC. We have $R^8_{min} = R^8_{max} = DC$. Consequently, bona-fide objects cannot be located at neighboring partition regions.

Let o_1 and o_2 be partition forming objects such that o_1 is of fiat kind and o_2 is of bona-fide or of fiat kind. Boundary parts of those objects may be co-located, i.e., their exact regions may be externally connected, EC. In terms of boundary-sensitive rough location we define: $F4(o_1, o_2) \equiv R^8_{max}((\text{loc}_{bs}\ o_1), (\text{loc}_{bs}\ o_2)) \leq EC$ and postulate $\forall o_1 \in F^\phi, \forall o_2 \in (F^\psi \cup BF) : (o_1 \neq o_2 \text{ and } r(o_1), r(o_2) \in G) \Rightarrow F4(o_1, o_2)$. Due to the underlying partition structure we are able to postulate that partition forming fiat objects cannot overlap, i.e., the largest relation that can hold between two partition forming bona-fide objects is EC.

---

[2] BF is a finite (but may be very large) set of things that count as bona-fide objects with respect to the definitions given by [SV99].

[3] Two objects, o_1 and o_2 are connected if they are not disconnected i.e., $(r(o_1), r(o_2)) \notin DC$.

[4] F^ϕ the set of fiat objects of kind ϕ in the sense of [SV99].

4.2 Built environments

In this subsection I use the constraints defined above in order to describe the components built environments (layout, path system, situations) formally.

The layout of a built environment is formed by a set of partition forming objects. Formally, it is a triple $L =< G, BF_G, F_G >$, where G a set of regions forming a regional partition, BF_G is a set of partition forming bona-fiat objects, and F_G a set of partition forming fiat objects such that the following holds to be true: (1) $\forall o_1, o_2 \in BF_G : \quad o_1 \neq o_2 \Rightarrow F3(o_1, o_2)$; (2) $\forall o_1 \in F_G, \forall o_2 \in BF_G \cup F_G : \quad o_1 \neq o_2 \Rightarrow F4(o_1, o_2)$; (3) $G = \{r(o) \mid o \in BF_G\} \cup \{r(o) \mid o \in F_G\}$; (4) $\bigvee G = \top$. These are formal versions of the partition axioms $P1$ and $P2$, where $\bigvee G = g_1 \vee g_2 \vee \ldots \vee g_n$, $g_i \in G$ and $\top$ is the universal region, U, without the exterior, EXT, of the environment.

The path system. Let $\Gamma^G = (V, E, h)$ be a directed version of the dual graph of the topological graph[5] of the regional partition, G[6]. Consequently, every vertex, v_i, corresponds to a partition element g_i and $h(e) = (v_i, v_j)$ refers to the boundary segment (g_i, g_j) 'looking' from g_i to g_j[7][8]. The path system of the layout, Γ^L, is a sub-graph [NC88] of $\Gamma^G = (V, E, h)$. The graph $\Gamma^L = (V' \subseteq V, E' \subseteq E, h')$ is defined such that the edges, $e' \in E'$, correspond to boundary segments of partition-forming fiat objects of non-barrier sort in direction (g_i, g_j). The vertexes V' are the vertexes joined by those edges. For details see [Bit99]. Consider the right part of Fig. 1. It shows the path system of the parking lot discussed in Section 2. The long grey bar on the main road is the stretched vertex corresponding to the partition region occupied by the main road. The bold solid lines represent edges corresponding to non-barrier boundary segments. The arrows along the edges show their direction. If there are edges for each direction then the arrows are omitted[9].

Path system and movement. Let $r_t(o)$ be the exact region of the object o at moment t, let $r_T(o)$ be the set of all regions at which o was exactly located within the time interval $T = (t_1, t_2)$, i.e., $r_T(o) = \{r_t(o) \mid t_1 \leq t \leq t_2\}$, and let $\bigvee r_T(o)$ be the sum of all those regions. Let $\Gamma^L = (V', E', h')$ be the path system of the layout L. A path within the path system from v_1 to v_2, $\Gamma^L_{v_1,v_2} = (V'', E'', h'')$, is a connected subgraph of Γ^L beginning at v_1 and ending at v_2. This path is a path for the object o, $\Gamma^L_{v_1,v_2}(o)$, if and only if: (1) $R^8_{min}((\alpha\,(\bigvee r_T(o))), (\alpha\,\bigvee\{v_i \mid h''(e'') = (v_i, v_j)\})) = NTPP$[10]; (2) $h''(e'') = (v_i, v_j) \Rightarrow R^8_{min}((\alpha\,(\bigvee r_T(o))), (\alpha\,v_i)) = PO$. This says that $(\bigvee r_T(o))$ overlaps all regions along its path (2) and that $(\bigvee r_T(o))$ is a non-tangential proper part of the sum of all partition regions along its path (1).

Situations. A situation in a built environment is a triple $S =< L, BF_S, F_S >$, where L is the layout of the environment, BF_S is a set of non-partition forming bona-

[5] See [NC88] and [Bit99] for details.

[6] Boundary sensitive approximations, $(\text{loc}\, o)$, correspond to labeled versions of this graph [Bit99].

[7] Multiple, disconnected boundary segments are distinguished by additional indices.

[8] In the remainder I use v_i and g_i synonymously.

[9] In this paper I only consider partition forming objects as *wholes*. In fact partition forming objects have parts which are caved out by fiat boundaries. A path system taking parts of partition forming objects into account is much better structured.

[10] Since the v_i refer to partition regions we have $R^8_{min} = R^8_{max}$.

fide objects and F_S is a set of non-partition forming fiat objects. The members of both sets are populating L in situation S. In a situation S the following holds: (1) $\forall o_1, o_2 \in BF_S \cup BF_G : \ o_1 \neq o_2 \Rightarrow F1(o_1, o_2)$; (2) $\forall o_1, o_2 \in F_S : \ (\phi o_1$ and ϕo_2 and $o_1 \neq o_2) \Rightarrow \ F2(o_1, o_2)$; (3) $\forall o \in BF_S \ : \ \Gamma^{L\cup\{EXT\}}_{r(EXT),r(o)}(o)$. Axioms (1) and (2) govern the non-partition forming objects as discussed in the sections 2 and 4.1. Consider axiom (3). The symbol EXT denotes the 'The world exterior to the environment L' and $\Gamma^{L\cup EXT}$ is the graph representing the path system of the environment L with its exterior EXT. Consequently, $\Gamma^{L\cup\{EXT\}}_{r(EXT),r(o)}(o)$ is a path for the object, o, from the exterior to its current location. Axiom (3) ensures that for each non-partition forming bona-fide object within the environment there exists a path along which this object could have been moved from the entrance to its current position. This is a formal version of the axiom M discussed in Section 2.

4.3 Specific built environments

In Section 2 we discussed that domain specific constraints on relations involving objects of different kind are weaker than constraints involving objects of the same kind. They can be violated without violating the laws of logic or physics, i.e., *it is possible to violate those constraints*. On the other hand the built environment *must permit* the satisfaction of those constraints in order *to be* an environment of a given kind.

Consider a parking lot with layout $L = (G, BF_G, F_G)$ and the informal axioms $S1$ and $S3$ as discussed in Section 2. Let $PS \subset F_G$ be the set of parking spots and let $BA \subset F_G$ the set of blocked areas of the parking lot. Let $CARS \subset BF_S$ be the set of cars populating the parking lot. We postulate: (1) $\forall o_1 \in CARS, \forall o_2 \in PS \ : \ \max\{R^8_{max}((\text{loc}_{bs}\ o_1), (\text{loc}_{bs}\ o_2)) \mid R^8_{max} \in RCC8\} = NTPP$[11]; (2) For each parking spot, $o_2 \in PS$ there must exist a path for a car $o_1 \in CARS$, i.e., $\Gamma^{L\cup\{EXT\}}_{r(EXT),r(o_2)}(o_1) = (V, E, h)$, which keeps blocked areas clear, i.e., $\forall e \in E \ : \ h(e) = (v_i, v_j) \Rightarrow \neg\exists ba \in BA : r(ba) = v_i$.

Axiom (1) states that cars need to fit into parking spots. Two remarks. Firstly, (1) is consistent with $\exists o \in CARS, \exists o_2 \in PS : PO(r(o_1), r(o_2))$, i.e., when we postulate that a parking lot *must be* such that cars do *fit* into parking spots we do *not* rule out the possibility that there are cars parked across boundaries of parking spots. Axiom (1) ensures the possibility for cars to be parked in parking spots. Secondly, stating axiom (1) in terms of rough location rather in terms of exact location has the advantage that we can effectively check its satisfaction since there are only finitely many different rough location in a built environment and we have the calculus proposed by [BS00] to compute the possible relationships.

Axiom (2) states that it must be possible for cars to avoid blocked areas. Again postulating this for an environment does not conflict with the fact that there cars that drive through or park at blocked areas.

[11] Since $r(PS_i) \in G$ we have $R^8_{max} = R^8_{min}$ and hence $\max\{rcc8(o_1, o_2) \mid rcc8 \in RCC8\} = NTPP$.

5 Conclusions

Given that task 1 and 2 are to be performed by program (II), there are three main arguments in favor of the formalization of build environments based on *rough location within environments* in opposition to the formalization based on *exact location of objects*: Rough location focuses on the relationships between objects and their environments; Concentrating on properties of the environment allows to abstract the different character of constraints on relations between the different kinds of objects forming and populating it; The notion of rough location is qualitative in nature.

Firstly. Rough location focuses on the approximate location of objects within regional partitions. In built environments the regional partitions formed by the partition forming objects are the main organizational structure. They provide a frame of reference within which non-partition forming objects are located. The notion of rough location implicitly takes the distinction between partition forming and and non-partition forming objects and the organizational structure of the regional partition into account.

When we describe built environments in terms of rough location then objects are second class citizens. The first class citizens are mappings representing the *rough location of objects within their environments*. These mappings can be interpreted as equivalence classes of objects sharing the same rough location. Since built environments are formed by finitely many partition forming objects there are only finitely many different rough locations within an environment. Given the calculus presented in [BS00] the satisfaction of the axioms presented in this paper can be checked effectively.

Secondly. Concentrating on properties that need to satisfied by the (built) *environment* allows to abstract from the different character of constraints on relations between spatial *objects*. The different character of the constraints on relations between objects is due to the fact that there are constraints that are enforced by the laws of logic, there are constraints that are enforced by the laws of physics, and there are constraints that are enforced by human conventions. The laws of logic prohibit objects of ontological same kind and partition forming objects to overlap. Laws of physics prevent bona-fide objects from being connected. Constraints involving fiat objects of ontological different kind are based on social rules and agreement and may be violated in certain situations. An environment *must permit* the satisfaction of *all* constraints in order *to be* an environment of a given kind *independently* of the character of the constraints between the objects forming or populating it.

Thirdly. We assumed a program (I) that generates potential plans for built environments. It is fair to assume that (I) is based on standard algorithms of computational geometry. The output of (I) is quantitative and focuses on metric knowledge. The program (II) extracts qualitative knowledge and builds a corresponding boundary sensitive rough location representation.

One might ask 'Why do we need a qualitative description if we have a quantitative geometric model?'. The answer is that it is the purpose of (II) to evaluate the plan of the environment with respect to axioms specifying what a plan of a built environment is AND with respect to the degree it facilitates human way finding. (i) In order to capture the essence of what a built environment is one needs to abstract from metric properties of particular instances. What a built environment is can be described in terms of (qualitative) relationships between ontologically salient features of the environment.

(ii) Human cognition is based on processing qualitative rather than quantitative knowledge. Qualitative knowledge about actual situations is based on observations of reality. Consequently, the question is not whether or not to use the quantitative description generated by (I), but to derive qualitative spatial relations *between ontologically salient features, which correspond to relations observable in reality* from this description. This is exactly what happens when we describe built environments in terms of boundary sensitive rough locations of objects forming and populating them.

In this paper I have shown that based on the notion of boundary sensitive rough location task (1) of program (II) can be performed, i.e., it is possible to decide whether or not a configuration of lines in the plane represents a built environment using the axioms given in Section 4. I, furthermore, showed how to derive paths within a build environments along which non-partition forming objects can move. This provides the formal foundations for task (2), i.e., to evaluate those paths with respect to the complexity of the way finding task to be solved in order to navigate along them. Subject of ongoing research in this context is to apply the model for the evaluation of the complexity of wayfinding tasks proposed by [RE98].

References

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Bit99] T. Bittner. The qualitative structure of built environments. Technical report, Department of Computer Science, Queen's University, 1999.

[BS98] T. Bittner and J. G. Stell. A boundary-sensitive approach to qualitative location. *Annals of Mathematics and Artificial Intelligence*, 24:93–114, 1998.

[BS00] T. Bittner and J. Stell. Approximate qualitative spatial reasoning. Technical report, Department of Computing and Information Science, Queen's University, 2000.

[CBGG97] A.G. Cohn, B. Bennett, J. Goodday, and N. Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *geoinformatica*, 1(3):1–44, 1997.

[CV95] R. Casati and A. Varzi. The structure of spatial localization. *Philosophical Studies*, 82(2):205–239, 1995.

[GLM83] T. Gaerling, E. Lindberg, and Maentylae. Orientation in buildings: Effects of familarity, visual access, and orientation aids. *Applied Psychology*, 68:177–186, 1983.

[Kui78] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–154, 1978.

[Lyn60] Kevin Lynch. *The Image of the City*. MIT Press, Cambridge, 1960.

[NC88] T. Nishizeki and N Chiba. *Planar Graphs: Theory and Applications*. North Holland, Amsterdam, 1988.

[RE98] M. Raubal and M. Egenhofer. Comparing the complexity of wayfinding tasks in built environments. *Environment & Planning B*, 25(6):895–913, 1998.

[Rou94] Joseph O' Rourke. *Computational Geometry in C*. Cambridge University Press, 1994.

[Smi95] B. Smith. On drawing lines on a map. In A.U. Frank and W. Kuhn, editors, *Conference on Spatial Information Theory, COSIT*, volume 988. Springer, Semmering, Austria, 1995.

[SV99] B. Smith and A.C. Varzi. Fiat and bona fide boundaries. *Philosophy and Phenomenological Research*, 1999.

[SW75] A. Siegel and S. White. *The development of spatial representations of large-scale environments*, volume 10, pages 9–55. Academic Press, 1975.

Experimenting NUMA for Scaleable CDR Processing

W.L.A Derks, S.J. Dijkstra, H.D. Enting, W. Jonker, J.E.P. Wijnands

KPN Research, P.O. Box 15000, 9700 CD Groningen, The Netherlands
{w.l.a.derks, s.j.dijkstra, h.d.enting,
w.jonker, j.e.p.wijnands}@kpn.com

Abstract. This paper describes the final results of an assessment of the scalability of the NUMA architecture for very large CDR databases[1]. First a description of the experiment case and experiment set up is given. Next the results of the experiments are described. The paper ends with a conclusion on the applicability of the NUMA architecture for very large CDR stores.

1 Introduction

The generation of so called Call Detail Records (CDRs) by modern telecommunication switches has opened the way to a whole area of new applications. Most importantly CDRs are the basis for almost all operational billing systems today. And recently, with the advance of database technology for very large databases, new applications based on the analysis of huge amounts of CDRs (tens of millions a day) have come within reach. Examples of such applications are traffic management, fraud detection, and campaign management.

Given the large amounts of data (terabytes of CDRs are normal) there is a need for very large and high performance databases. One way to accommodate these requirements is to use specialised high performance database servers such as for example IBM mainframes, Compaq (TANDEM) or TERADATA machines. However, telecommunication companies are hesitating to make large investments in technologies supporting these new applications. What they are looking for is cost-effective and scaleable technology for realising very large databases in an incremental way. Given that there is a general trend of commodity hardware and software becoming more and more powerful with respect to both processing power and storage capacity, the question arises whether this technology is mature enough to support the specific CDR analysis applications of telecommunication companies.

To answer this question we performed a state-of-the-art study on scaleable architectures for CDR stores (see [1]). There we concluded with two promising solutions: NUMA and NT clusters. Following the state-of-the-art study, we experimented with both technologies to assess the scalability[2] of both solutions. This paper describes the results of the experiments with a CDR store with Oracle 8.0.5 on IBM NUMA-Q 2000[3]. Results of experiments with DB2 UDB EEE on an NT cluster will be published in an other paper later on.

---

[1] The work described here was carried out in the context of EURESCOM project P817 “Database Technologies for Large Scale Databases in Telecommunication”.

[2] Remark that we did not perform a benchmark.

[3] In 1999, IBM acquired Sequent Computer Systems to get hold on the NUMA technology.

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 970–980, 2000.

2 Case Description

As case for performing the experiments, we took the Marketing Customer Data Base (MCDB) of KPN Mobile as an example. This data warehouse CDR store contains the CDRs and customer data of all customers of KPN Mobile. The system only has 1 to 5 concurrent users with very heavy marketing and management queries.

2.1 NUMA Architecture

NUMA stands for *Non-Uniform Memory Architecture.* A NUMA system consists of multiple *Symmetric Multi Processing* (SMP) nodes that share a common global memory. The nodes are interconnected by a very fast interconnect. As the system uses a single shared memory and a single instance of the operating system, the system looks like a normal SMP system but with far better scalability properties. The system we investigated was a NUMA-Q 2000 System from IBM running a proprietary, System V based, operating system DYNIX/ptx. An overview of the architecture is shown in fig. 1. The NUMA architecture can scale up to 16 Quads, but together with IBM we decided to use at most four Quads for our database size. The system consists of two parts: the group of Quads on the left and the I/O sub-system on the right. Each Quad is a processing module with four Intel CPU's (Pentium 200 MHz), 1 Gbyte of main memory and two I/O cards (so-called firefly cards). A Quad has access to main memory of a remote Quad via the IQ-link; hence makes the system behave NUMA.

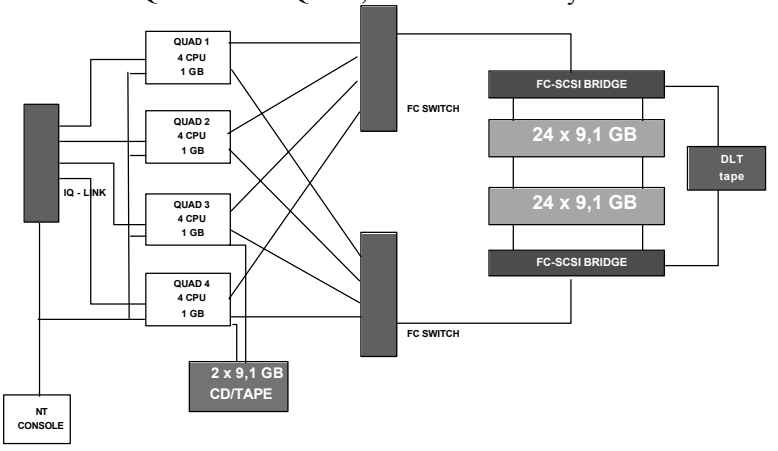


Fig. 1. Used IBM NUMA-Q 2000 architecture

A Quad is connected to the I/O subsystem via fibre channel. As the disks are still SCSI based, the fibre channel protocol is converted to SCSI by the FC-SCSI bridges. Each FC-SCSI bridge is connected to 24 disks of 9,1 Gbyte each. Each group of 12 disks is pair-wise combined in a so-called PBay. As all disks are mirrored, each PBay delivers a net storage capacity of 6 times 9,1 Gbyte making 54,6 Gbyte. However, the last disk of each six pack is reserved for purposes like archiving, file system and hence is not included in the database storage. This leaves us 45,5 Gbyte for each PBay. Our system contained four PBays, which provided us with a net storage capacity of 182 Gbyte over 40 disks (the gross storage capacity is 360 Gb).

2.2 Database Schema

The NUMA system was investigated primarily for its data warehouse potential. The data we used included a large CDR table and detailed customer data. The customer data included name, address, and detailed product, order and delivery information. The CDRs occupied in total 90% of the total data volume and only 10% of the data were the actual customer data.

2.3 Query Load

The scalability is investigated by executing diverse queries against different system configurations. The queries executed (nine different) all included real-life marketing queries, most of them a mix of join and group-by operations. Note that we investigated read-only queries and focussed on intra-query parallelism. During the experiments, two query groups showed similar behaviour. From both groups we picked one representative query: [Group by] and [Multi-way Join]. We feel that both queries are representative for the system scalability and we will discuss these queries in more detail in this paper. In the rest of this paper we will refer to these queries as GB (Group-by) and MJ (Multi-way Join).
The GB query answers the business question: ”What is the total call duration between two connections (servmobnr and otherprtynr) in the so-called peak-hours (between 07:00 a.m. and 08:00 pm)?” The SQL statement we build for this question is:

```
SELECT servmobnr, otherprtynr, sum(to_number(calldur))
FROM cdr
WHERE to_char(strtchrgdate, 'hh24') between '07' and '20'
GROUP BY servmobnr, otherprtynr
```

The MJ quey answers the business question: “What is the total call duration per user?” This query is complex and the SQL statement is given here:

```
SELECT ads.achternaam, sum (to_number (calldur))
FROM
 mcdb_leveringen lvg, mcdb_klanten klt, mcdb_adressen ads, cdr,
 mcdb_individuele_produkten ipt
WHERE
cdr.servmobnr = ipt.kenmerk AND
 ipt.lvg_id=lvg.id AND
 cdr.strtchrgdate between lvg.datum_begin and lvg.datum_einde and
 lvg.klt_id=klt.id AND
 ads.klt_id=klt.id AND
 ads.tas_code='bezoe'
GROUP BY ads.achternaam
```

3 Experiment Setup

3.1 Approach

The primary aim of the experiments is to reveal the actual scalability behaviour of a NUMA architecture *in practice*. Therefore we configured the system initially with a small database (40 Gbyte) with moderate system power (1 Quad) and extended the database and system eventually towards 160 Gbyte with 4 Quads. To explain the measured system behaviour, additional experiments were performed to reveal the

impact of specific system resources on the scalability behaviour (i.e. cpu, memory, IQ-link). In total we performed about 300 experiments.

3.2 DBMS Configuration

As mentioned earlier, we used the Oracle 8.0.5 DBMS. The version we used was made NUMA-aware, which means that it is optimised for usage on a NUMA system, e.g. for local memory reference. Although we have multiple Quads, we did not use the Oracle Parallel Server (OPS) because OPS doesn't support intra-query parallelism. We left the configuration of Oracle's parameters in most cases default.[4]
The CDR table is a horizontally partitioned table with 12 partitions (a partition for each month).[5] All customer data tables and their indexes reside in two tablespaces. When possible, data and indexes were stored on different PBays. The database size is scaled from 40 GB to 80 GB to 160 GB. In these figures both DATA, INDEX and TEMPORARY space are included. As the database is mirrored, the total disk space used is 80, 160, and 320 GB. The different database sizes were spread over the four mirrored Pbays. The smallest database size DB40 was spread over two Pbays and DB80 and DB160 were spread over all four Pbays. Inherently, more disks were available for the DB80 and DB160 experiments (see Fig. 2). We feel that this database storage expansion is quite realistic.

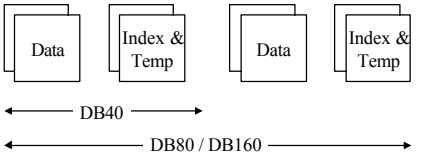


Fig. 2. data placement

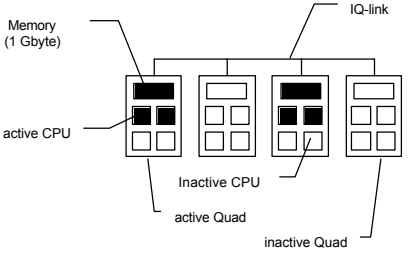


Fig. 3. NUMA configuration

3.3 NUMA Configuration

The NUMA system can be configured in different ways. Fig. 3 shows a schematic representation of a four-Quad NUMA system. Each Quad is equipped with four CPU's and one memory unit. A black/white color indicates switched on/off. We can change the number of quads, the number of CPUs per quad, and the number of disks. If the database size is 80 GB the configuration description of this example is P22DB80, which means that the configuration consists of 2 Quads, each with 2 processors switched on, each 1 GB of memory and a database size of 80 GB. The default value of the memory is 1 GB per quad, but in some experiments we kept the *total* amount of memory over all Quads constant at 1 GB. For example, the configuration P1111DB160M256, implies a 4 Quad setting, each with 1 processor switched on, 256MB memory per Quad at a database size of 160 GB.

---

[4] Notably, we worked with db_blocksize of 16 KB and scaled db_block_buffers, sort_area_size and hash_area_size proportionally with the amount of memory available. Also, we worked with Parallel_max/min_servers set to 32.

[5] This was a manageability requirement.

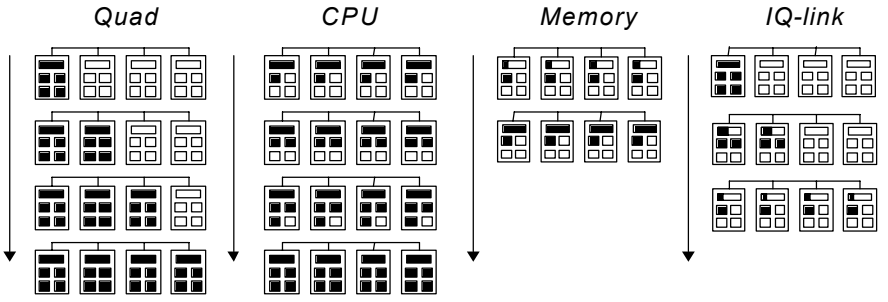


Fig. 4. Schematic overview of the experiments

To assess the behaviour of the system under this workload, we first examine the scale-up behaviour. Scale-up means that we scale the hardware and database proportionally. Ideally this would mean, that the response times of the queries remain constant. For a more in-depth analysis, we investigate the speed-up of the NUMA system and workload speed-down of the database system. The speed-up is investigated by expanding the system capacity, while keeping the amount of data constant. Ideally, the query response time should then decrease proportionally with the system power. The workload speed-down includes experiments with constant hardware configurations, but with an increased database size. Ideally, here we would see proportional increase of query response time with the size of the data volume.

3.4 Scale-Up

The scale-up configurations represent the realistic business setting: adding more Quads, as the business requires more processing power (increase in database size). Each step the number of Quads is doubled with its full power (four CPU's and 1 GB of memory) and the database size is doubled. These configurations should give a realistic scale-up impression. The configurations for the GB and the MJ experiments used for measuring the scale-up are: P4DB40, P44DB80 and P4444DB160.

3.5 System Speed-Up

To test in what perspective NUMA is responsible for this scale-up behaviour, we have investigated system speed-up. The speed-up experiments were performed by increasing system power (1, 2, 3, 4 Quads) at the database size DB160. Note that the amount of memory as well as the amount of CPU's increases with the number of Quads. The configurations used for measuring the system speed-up are: P4/44/444/4444DB160.

3.6 CPU Speed-Up

To investigate the effect of the CPU, we have performed experiments with four Quads, constant memory, but with CPU's turned on in stages: 4, 8, 12 and 16 CPU's. The configurations for measuring the CPU speed-up are: P{1111|2222|3333|4444}DB{40|80|160}.

3.7 Memory Speed-Up

Expanding the system with Quads tests the effect of the memory scaling. As we want the processing power to be equal, the amount of CPU's is kept constant. Note that the

processors do not have to access remote memory. The different configurations are: P1111DB{40|80|160}DB160 and P1111DB{40|80|160}DB160M256.

3.8 Distribution Effect

Of primary interest to us is the behaviour of NUMA in a data warehouse environment, which translates itself into the investigation of the performance penalty of remote memory access versus local (Quad-internal) memory access. We investigate this by first enabling four CPU's on one Quad and then spreading the CPU's step-wise over the other Quads: P4DB{40|80|160}, P22DB{40|80|160} and P1111DB{40|80|160}.

3.9 Workload Speed-Down

Now we have investigated the speed-up behaviour, we analyse the scalability behaviour of the queries. This way we can identify to what extent the used algorithms of Oracle are responsible for the scale-up behaviour. The hardware configurations in the different settings are the same. The size of the database varies. The different configurations used for measuring the workload speed-down are: P4444DB{40|80|160}.

4 Results

As the query plan has significant effect on the query response time, we first verified Oracle's access plan. For all queries it appeared that the query plan was equal for all system configurations. The query plan for the GB query was:

```
SELECT STATEMENT
 GROUP BY A1.C0, A1.C1
    PARTITION CONCATENATED
     TABLE ACCESS FULL CDR
```

The query plan for the MJ query was:

```
SELECT STATEMENT
 SORT GROUP BY
  HASH JOIN
   HASH JOIN
    MERGE JOIN
     SORT JOIN
      HASH JOIN
       TABLE ACCESS FULL MCDB_INDIVIDUELE_PRODUKTEN
       TABLE ACCESS FULL MCDB_LEVERINGEN
     FILTER
      SORT JOIN
       PARTITION CONCATENATED
        TABLE ACCESS FULL CDR
    TABLE ACCESS FULL MCDB_KLANTEN
   TABLE ACCESS FULL MCDB_ADRESSEN
```

Note that no index access was used, although we had defined indexes on the relevant fields. All tables were accessed by a full scan.

4.1 Scale-Up

The scale-up measure indicates how well the NUMA system can cope with increasing amounts of data. Ideally, the system should keep pace with the increasing amount of data stored in the system. Hence, doubling the amount of system power should compensate for doubling the amount of data. How these figures relate in practice is shown in Fig. 5.

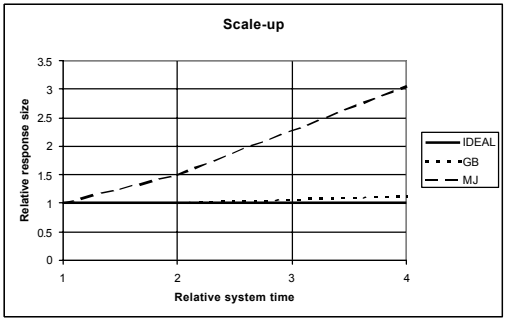


Fig. 5. Scale-up of CU, GB, MJ, MQ1 and MQ2

When the amount of data stored in the database is kept the same, we see that the GB query keeps the same relative response times when expanding the data and system power. However, the response times of the MJ query increases significantly. Apparently, the NUMA hardware cannot compensate for the increased MJ query complexity when expanding the whole system linearly. As the ratio amount of memory per database size remains constant, the deviation cannot be explained by the switching from main-memory to TEMP (read: disk) processing. For the MJ query we have identified increased reads while expanding the database, which show non-linear scaling. Unfortunately, we were not able to identify the effect of the I/O system exactly. As we did not scale the I/O system, the I/O system will become increasingly determinant on the scalability behaviour of the system as the Quad configuration is improved. However, this does not explain the bad scaling of the MJ query in the first scaling step from one Quad to two Quads, because the I/O system was not a bottleneck.

4.2 System Speed-Up

To test in what perspective NUMA is responsible for this scale-up behaviour, we have investigated system speed-up. The speed-up experiments were performed by increasing the system power (1,2,3 and 4 Quads) for each of three database sizes DB40, DB80, and DB160. Note that the amount of memory as well as the amount of CPU's increases with the number of Quads. The relative response times[6] are included in Fig. 6.
We see that the speed-up of the queries improves with the size of the database, i.e. closer to the IDEAL line. Apparently, queries over large databases benefit significantly from extra Quads. To figure out what resource causes the speed-up to improve, we determine what parameters are involved in the three pictures above. The components we investigated include: CPU, Memory and IQ-link.
The I/O system has large influence on the scalability behaviour of the system. Unfortunately we did not have time nor money to include this parameter into our

[6] Relative means that the response times are displayed relative to the first response time.

experiments. Therefore we kept the I/O system parameter as constant as possible during the experiments (see Fig. 3). We feel that the figures are still representative for the NUMA scaling characteristic.
Next, we investigate the impact of CPU, Memory and IQ-link on scalability.

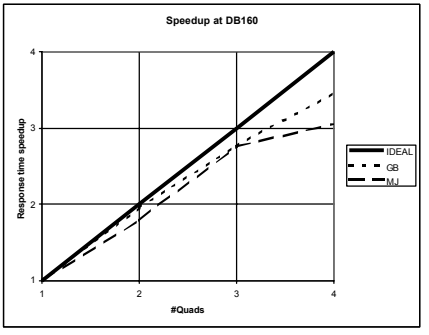


Fig. 6. Quad speed-up

4.3 CPU Speed-Up

To investigate the effect of the CPU, we have performed experiments with four Quads, constant memory, but with CPU's turned on in stages: 4, 8, 12 and 16 CPU's. The results are shown in Fig. 7.

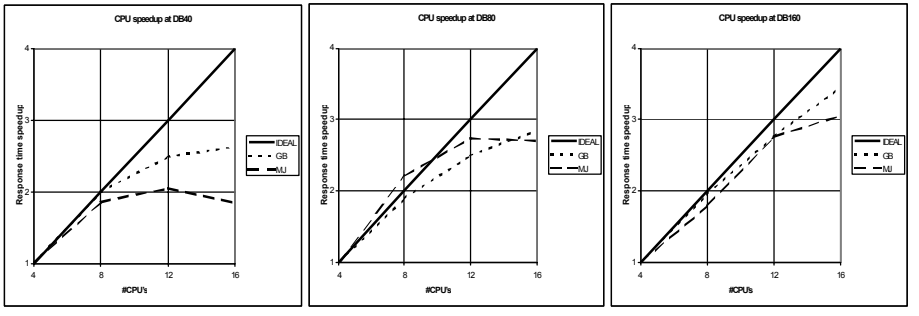


Fig. 7. CPU speed-up

The scalability of GB is very sensitive to CPU resources, which can be explained by its large CPU intensive sorting operation. The MJ query is less sensitive to CPU. This is especially so for large database sizes, where GB scales on near linear even at 16 CPU's, whereas MJ has enough at 12 CPU's. The I/O system was kept constant, thus inherently this resource became increasingly important when we scaled along with Quads. Especially for smaller database sizes the I/O system fell short quite soon. Mainly because of this, the scalability figures at DB40 show deviations from linear quite soon.

4.4 Memory Speed-Up

We expected memory to have an important influence on query performance. However, Fig. 8 shows that memory does not have significant impact, especially for large database settings. This goes certainly for the GB query. The MJ query benefits from more memory only at small databases, but at DB160 there is no benefit anymore.

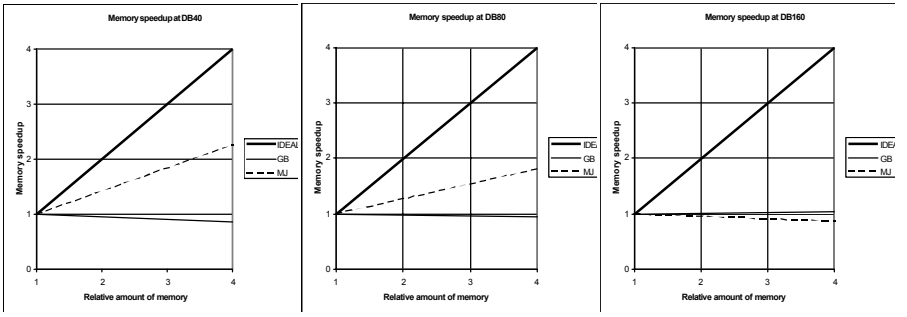


Fig. 8. Memory speed-up

This can be explained by the fact that queries on large databases always have to write to TEMP, because pure in-memory processing is not possible. The impact of extra memory is only relevant when it can preserve in-memory processing. In that case, the difference can be significant. For the MJ query we see, that additional memory can help for smaller database sizes. This is explained by the fact that for the small memory setting Oracle performs sorting via TEMP, but with the extra memory a pure in-memory sorting strategy can be adopted. This is beneficial for performance. At DB80 Oracle can still adopt a complete in-memory strategy for the largest memory, while at the small memory size more data has to be written to disk. At DB160, the larger memory cannot contain all data anymore in memory. Because at DB160 Oracle must use TEMP as well, the advantage of the extra memory is not significant anymore. Concluding, extra memory is only significant, when in-memory processing can be preserved. Extra memory is not significant when TEMP has to be accessed anyway. This is the case for very large data sets.

4.5 Distribution Effect

The spread-out of processing across multiple Quads is the typical characteristic of NUMA. Therefore we explicitly investigated the processing of a query on a single Quad, two Quads and four Quads, while maintaining the number of CPU's, the amount of memory and the database size constant. Hence, we could identify the impact of the IQ-link. Ideally, the response time should be the same for each configuration. From Fig. 9 we learn that the GB query is relatively insensitive to the distribution. However, the MJ query shows increased response times when adding more Quads. Detailed analysis revealed, that the MJ query suffers from significant CPU utilisation degradation. CPU utilisation drops from 100% at a one Quad to 72% at a four Quad configuration . User time even drops from 90% down to 62%. Initially, we thought that the NUMA IQ-link would cause this overhead. To determine whether this is a NUMA or Oracle issue, we consulted both companies. According to IBM remote memory access across the IQ-link is a few times the local memory access time. As the difference is this small, it is not worth for the OS to put a process on wait while fetching the remote pages, because this would require a kernel switch and a kernel switch takes typically some milliseconds. Therefore the machine lets the CPU process some wait cycles before it proceeds. As wait cycles from the CPU occur in user time, the degradation of user time to 70% cannot be explained from the NUMA IQ-link hardware. However, it should be noted, that the Quads provide optimal performance

with four active CPU's. Decreasing the number of CPU's could imbalance the Quad performance and lead to sub-optimal results.

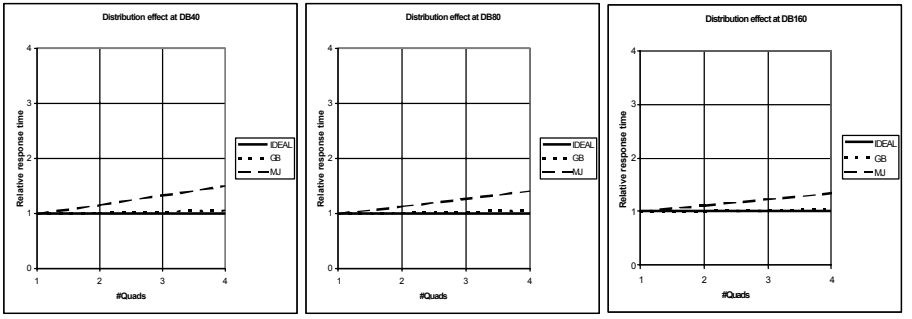


Fig. 9. Distribution effect at DB160

We cannot conclude other than that Oracle is responsible for the decreased CPU utilisation. The Oracle expert we talked to did not deny this, but could not confirm it either. In addition to MJ, other queries appeared to be sensitive to the Quad distribution as well.

4.6 Workload Speed-Down

Finally, we identified the influence of the Oracle algorithms on the scale-up behaviour of the system. The results are shown in Fig. 10.

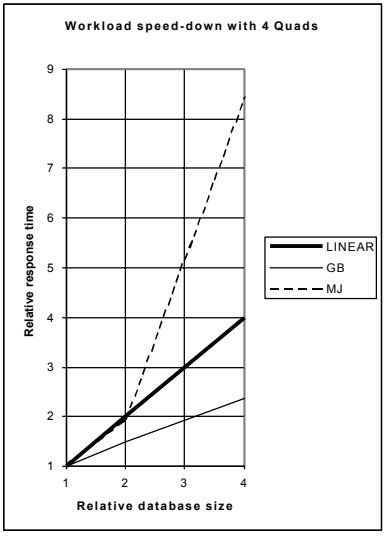


Fig. 10. Workload speed-down for 4 Quad configurations at DB40, DB80 and DB160

The speed-down of queries shows straight behaviour for the GB query, although it is much better than linear. However, MJ shows size-sensitive behaviour. The explanation appears to be that the increased database size causes the query to run out of main-memory and hence results in costly TEMP processing. When we look back at the scale-

up chart (see Fig. 5) we see this jump again at the two Quad configuration. Hence, we see that the workload speed-down is rather significant in the scale-up characteristic.

5 Related Work

A few months after we finished our NUMA experiments, BT, Oracle, Sequent (IBM) and EMC^2 came with the successful results of a so called Extremely large Proof of Concept (EPoC) for a BT 35 Tb data warehouse on a NUMA machine (see [5]). The goal of this test was to proof that such a extremely large data warehouse can be build (with the right partners) and can work with NUMA whereas we focussed on the actual scalability and manageability issues. In [6], the processing of multi-way join queries on a large database in both shared-nothing (cluster) and shared-memory (NUMA) is investigated. An alternative execution model for NUMA is proposed to overcome problems caused by skew of data. An overview of research activities on more fundamental architectural NUMA, and related COMA, issues can be found in [2]. In [3], we are supported in our observation as the Gartner Group also spots NUMA and Clustering as promising technologies to overcome the limitations of the SMP architecture without the burden of the MPP architecture. In [4], the clustering of PC's versus workstations is examined. In that work, the focus is on scientific and engineering computations rather than database processing.

6 Conclusions

Scalability is preserved by NUMA's modular architecture. Adding new Quads provides always the opportunity to install state-of-the-art technology, even after years. This is an advantage in relation to SMP systems, because they often require a homogeneous CPU setting. Speed-up of the NUMA system is good on average. For scaling up the database the CPU power appears to be the most important resource with respect to NUMA. Sorting queries provide very good scaling, which is due to Oracle's (Quad-)local sorting strategy and the CPU intensity of relational sorting. Problems arise with multi-way join processing. An important factor here is the bad scaling of the query with increased database size (workload speed-down), but seems not to be NUMA specific. In addition we have identified some performance degradation due to the distribution of the query processing across multiple Quads. This is NUMA specific.

References

1. EURESCOM, Project P817, "Database technologies for large scale databases in Telecommunications", Deliverable 1, Overview of Very Large Database Technologies and Telecommunication Applications using such Databases (March 1999)
2. Wain, J., *CC-NUMA bibliography*, http://www-frec.bull.com/docs/biblio.htm, (October 1996)
3. Dolan, D., *SMP versus NUMA versus Clustering: Which architecture is best?*, Gartner Group Market Analysis (December 1998)
4. Carter, R. Laroco, J., *Commodity clusters: Performance comparison between PC's and Workstations*, Proceedings of HPDC (May 1996)
5. BT, Oracle, Sequent, EMC^2, *BT EPoC Project Report* (December 1999)
6. Bouganim, L. et. al., *Load Balancing for Parallel query Execution on NUMA Multiprocessors*, Distributed and Parallel Databases 7, 99-121 (1999)

SPICE: A Flexible Architecture for Integrating Autonomous Databases to Comprise a Distributed Catalogue of Life

Andrew C. Jones[1], Xuebiao Xu[1], Nick Pittas[1], W. Alex Gray[1], Nick J. Fiddian[1], Richard J. White[2], John Robinson[2], Frank A. Bisby[3], and Sue M. Brandt[3]

[1] Cardiff University, Department of Computer Science, PO Box 916, Cardiff CF24 3XF, UK
{Andrew.C.Jones|X.Xu|N.Pittas|W.A.Gray|N.J.Fiddian}@cs.cf.ac.uk
[2] Biodiversity & Ecology Research Division, School of Biological Sciences, University of Southampton, Southampton, SO16 7PX, UK
{R.J.White|J.S.Robinson}@soton.ac.uk
[3] Biodiversity Informatics Laboratory, Centre for Plant Diversity & Systematics, The University of Reading, Reading RG6 6AS, UK
{F.A.Bisby|S.M.Brandt}@reading.ac.uk

Abstract. In the SPICE project we are building a distributed catalogue of life, which will eventually be formed from up to 200 autonomous taxonomic databases. We are faced with a number of challenges, which include the scalability of the system; the accommodation of partial or missing data; queries which are potentially very expensive computationally, where it is difficult to determine which databases will contain data matching the queries, and the effective integration of heterogeneous databases at the knowledge level. In this paper we present the architecture on which SPICE is being built, and we explain how, within our SPICE architecture, we will be able to explore and develop new techniques to enhance access to the SPICE distributed database.

1 Introduction

The Species 2000 project has as its goal the creation of a catalogue of all known species of living organisms. It is intended that the data held in this catalogue should be of high quality and a complete coverage of species should be achieved. It has been explained elsewhere (e.g. [3, 4, 6]) why it is important to have such a catalogue of life available. Here, we address the problems which we face in seeking to build such a resource, particularly technical problems, and explain how we are seeking to solve them. Accordingly, we shall concentrate on the SPICE (Species 2000 Interoperability Co-ordination Environment) project. SPICE is allied to Species 2000 but with the specific brief of researching the Computer Science challenges associated with building a scalable infrastructure for Species 2000.

One of the major tasks performed by taxonomists is to classify organisms into *species*. The essential problem we face is that no taxonomist has a monopoly

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 981–992, 2000.

of taxonomic knowledge: a taxonomist will typically specialise in one particular group of organisms. Moreover, where the expertise of taxonomists overlaps, there is room for variation of opinion. Taxonomists sometimes build databases of the species with which they deal, and the strategy being adopted by Species 2000 is to select suitable databases from those available in order to form the basis of a composite distributed taxonomic database, comprising individual *global species databases* (GSDs) that are chosen to be broadly non-overlapping in taxonomic coverage [3]. The idea that there should be such a federation of GSDs was already specified in the Species 2000 project before the SPICE project commenced. Where possible, we are selecting existing GSDs rather than creating new ones. This is because of the large amount of effort that taxonomists have already expended in building GSDs of high data quality. Since they have been built independently, we are faced with the problem of heterogeneity. To be successful, we need a means of making these distributed, autonomous heterogeneous databases available on-line as a tightly coupled federated database system, without compromising the autonomy of the individual databases in any way. Owners of such GSDs must be free to update their databases whenever their understanding of the taxonomy of the organisms in question is revised. Moreover, it must be relatively straightforward to bring new GSDs into SPICE, even if they do not have the same schemas as any of the existing GSDs or they are not built on a familiar platform. Indeed, we have discovered by consultation with potential GSD suppliers that there is a wide variety of preferred technologies for joining their GSDs to SPICE: some wish to use ODBC; some CGI, etc., due to their previous skills and experience. Moreover, there is variation in the schemas: e.g. the queries to retrieve the relevant information on a single species will vary quite substantially. It is not the role of SPICE to do all the work necessary to bring each individual GSD into the system, but to facilitate the building of suitable *wrappers*. Diversity of wrappers is therefore required.

We focus in this paper upon how the SPICE system has been designed so that we can explore how to meet these requirements, and upon the kinds of techniques we plan to develop and how their effectiveness will be tested. More generally, SPICE is an example of dealing with a (slowly) changing taxonomy – here, in the strict biological sense – where the taxonomic knowledge is distributed.

1.1 The Scenario

As a species information system, the idea of Species 2000 is that a user should be able to type in a (possibly partially-specified) species name – either the *scientific name* or the *common name* for the species – and retrieve a list of matching species. Because of the nature of such names, this list may occasionally comprise species coming from more than one GSD. The initial goal is that the user should then be able to retrieve seven pieces of information on any given species:[1]

1. Accepted Scientific Name, with reference citation(s) (Obligatory);

[1] Further detail of these data items is given at the Species 2000 Web Site, http://www.sp2000.org

2. Synonyms, each with status and reference citation(s) (Obligatory where appropriate);
3. Common Names, each with country, language and reference citation(s) (Optional);
4. Latest taxonomic scrutiny (Optional);
5. Source database, including short name, version number, date and URLs (Obligatory);
6. Comment field chosen by the GSD co-ordinator (Optional), and
7. Family (Obligatory).

This is the essential information that a biologist will need to know in order to consult the literature or to consult other on-line sources in regard to a particular species. The URL mentioned above provides a link to any additional information stored in the GSD, which may be browsed by the user as required. In addition to this, another goal is eventually to develop the Species 2000 system so that when a species is located in a GSD, GSD owners will be encouraged to arrange for their Web pages to support searching of other biological databases for further information about the chosen species. It may therefore be seen that wrapping to a fixed schema is appropriate: there is heterogeneity in the component database schemas and most of them contain extra information, which may be used in generating the Web pages, but it is not otherwise required by SPICE. We therefore concentrate on accommodating the variation in the ways that the seven items listed above are represented in the individual schemas.

1.2 Paper Outline

In the remaining sections, we mention first some other attempts that have been made to develop global catalogues of life, and to exchange taxonomic information. We discuss how our work relates to other projects that seek to achieve interoperability among heterogeneous databases. We then discuss the SPICE architecture. First we present some early systems, implemented as 'proof of concept', which do not fully meet these requirements. Then we present the current SPICE architecture. We explain the benefits offered by this architecture in supporting the kinds of heterogeneity with which we need to deal, and show how it provides a practical basis for developing techniques to address the challenges that we face. The essential challenge is how the system can continue to operate efficiently when a number of users are simultaneously making queries over a distributed database comprised of a large number of individual heterogeneous databases. We also need to provide an acceptable quality of service when GSDs are unavailable for a period (or even, in some cases, do not yet exist). We then return to evaluate our achievements so far and identify areas needing continuing research.

2 Background

One could envisage various ways of realising a global catalogue of life, but there is good reason to believe that the approach taken by Species 2000 is the most

likely to succeed [3, 4, 6]. We have already indicated that our catalogue of life is a distributed resource, comprising heterogeneous taxonomic databases. In this section we discuss previous research relating to the building of (i) a catalogue of life, and (ii) interoperable systems. We shall explain that the other, centralised approaches to building catalogues of life have inherent difficulties, and that there are well-established techniques in building interoperable systems that are applicable to SPICE. Note, however, that the restrictions we impose on the data model and possible queries mean that we have a suitable application to explore and experiment with techniques for improving the operational efficiency of the system, and the efficiency with which it can be built. As indicated above, solutions to these problems comprise the main achievement that will be required within the SPICE project.

2.1 Related Work in Taxonomy

The main related work in the field of taxonomic information systems may be divided into two:

- attempts to build centralised and/or homogeneous catalogues; and
- attempts to define standards for exchange of taxonomic data, suitable for use in a digital library context.

Two examples of centralised and/or homogeneous catalogues are the Integrated Taxonomic Information System (ITIS)[2] and the Tree of Life project [11]. In ITIS, a single database of taxonomic knowledge is being built, with responsibility for the taxonomic quality and integrity of the database distributed among a number of suitable experts. The Tree of Life project, in contrast, comprises a distributed set of HTML pages, interlinked in accordance with the evolutionary relationships held to exist between the taxa they represent. Again, responsibility for the knowledge represented is distributed among a number of experts.

The problem with both these systems is that the taxonomist does not have control over the data that is included, expressed in a form of his or her choice. Many taxonomists are using proprietary databases built using software such as Microsoft Access, with schemas that suit their individual needs. It is more desirable for taxonomists to maintain such databases themselves, and for us to integrate them into a catalogue of life automatically in some way. We shall see later that this is indeed how SPICE works.

Another potentially relevant project is the Z39.50 Biology Implementers Group (ZBIG) project[3], in which a standard for exchange of data within a Z39.50 [1] architecture is being developed. This 'Darwin Core', which is a specialised application of Dublin Core [18], could clearly play a role in SPICE: taxonomic databases would need to be wrapped or transformed so as to supply and accept data in Darwin Core format. But as the standard is still under development and we have chosen not to adopt a Z39.50 architecture (see below), the usefulness of this standard to us was judged to be limited.

[2] http://www.itis.usda.gov
[3] http://chipotle.nhm.ukans.edu/zbig/

2.2 Related Work in Interoperability and its Relevance to SPICE

Interoperability among heterogeneous databases has long been seen as being of major importance (e.g. [9, 15]). Various kinds of federated architectures are employed, depending on the nature of the databases involved. In particular, *wrapping* techniques are often used to remove some elements of heterogeneity (e.g. [14]) and *mediators* are used to draw together information from disparate sources (e.g. [19]). If one does not map to a common data model, extensive use of metadata is required in the interpretation of data. An increasingly common way of providing this is by use of XML (e.g. [2]). However, in SPICE, the scenario presented in Sect. 1.1 clearly imposes a set of requirements on the kinds of data model that the individual GSDs can have. In particular, there is a firm definition of the data that must be available to SPICE, and so it is natural to specify a common data model requiring all wrappers to map between the GSDs and that common data model. This has some bearing on the relevance of metadata standards to SPICE, as we shall see later.

Much work in digital libraries has been carried out using, and developing, Z39.50. The difficulties with adopting Z39.50 as the basis for SPICE are that (i) we wish to make use of various kinds of knowledge within the SPICE mediator itself, e.g. a taxonomic hierarchy, which does not fit in well with the gateway-to-databases model that characterises Z39.50; and (ii) more generally, we wish to select an architecture that does not *preclude* achievement of good performance (of course), but also it must allow us to experiment with novel techniques for achieving the required performance. We further note that there is increasing interest in alternative digital library architectures and, in particular, the Stanford Digital Library project [13] is built from distributed objects using CORBA [17] and provides interoperation between various kinds of services including Z39.50, HTTP and Telnet.

3 Architecture for SPICE

Before presenting the SPICE architecture to which we have now committed, we shall mention briefly four exploratory experiments that were carried out by the SPICE team and, previously, by the larger Species 2000 project team. This will provide the background to our decisions about the SPICE architecture we have adopted.

3.1 Prototypes

Four major experiments were carried out: a CGI-based system; an XML-based system, and two CORBA-based systems.

The CGI-based system[4] was built by D. Gee and J. Shimura, prior to the commencement of SPICE. It is designed so that the user types in the scientific name to be searched for, selects the database to be searched, and then submits

[4] Currently available at the Species 2000 Web site – http://www.sp2000.org

the query. Each GSD has an associated World-Wide Web server, to which the query is submitted using the CGI protocol, and a frame is dedicated to each database in order to display the HTML page that is generated, containing the results. Although this works well as a proof-of-concept, in particular illustrating to biologists the simultaneous use of distributed databases, it is inherently unscalable. This is because (i) there is a physical limit to the number of frames that can sensibly appear on one page and (ii) there is no mechanism to optimise searching, e.g. by automatically selecting relevant databases, or to control the combination of results. The GSD owner is responsible for the appearance of the HTML page that contains the results for his or her database. On the other hand, a suitable CGI format for the queries that the user will need to issue *has* been devised, as part of this experiment: this is something we exploit in our current SPICE system, as we shall see presently.

The XML- and one of the CORBA- based systems are both developments of an initial Java-based prototype in which the objects appropriate to SPICE were identified. (These are essentially the same objects identified in the next section.) In this Java prototype, a prototype wrapper was built that used the protocol of the previous CGI prototype to extract data from a single participant GSD, namely the International Legume Database & Information Service (ILDIS) [5]. But unlike the CGI prototype, when the HTML page is returned by ILDIS, the relevant information is *extracted* from it and displayed in a proprietary manner. It will be appreciated that as these HTML pages were never intended to be used in this way, extraction of such data is difficult and the software is vulnerable to changes of layout, as in any 'screen scraping' or similar application. But the feasibility of using CGI, where necessary, was established if it could be accompanied by a more accessible return format than had previously been used.

The XML prototype was built to explore the potential of XML, and of associated technologies such as the Persistent Document Object Model. It was discovered that although, unsurprisingly, it is possible to model the required taxonomic data using XML, it is not *necessary* within the 'hub' of the system. This follows since we do not need to exploit one of the most attractive features of XML, that documents are 'self-describing': we have a common data model.

The SPICE CORBA prototype required us to consider in principle what the common data model would look like, as the IDL (interface definition language) for the various objects in the system had to be defined. This was the second CORBA-based prototype: an earlier prototype had been built in Japan around a rather different architecture from ours [16].[5] The SPICE CORBA prototype showed that appropriate IDL could readily be defined (using *structs* for efficiency reasons, rather than objects, to convey taxonomic data). We thus opted for CORBA as the basis of the full SPICE system, because although it was not fully demonstrated in the prototype, CORBA provides flexibility in dealing with the kinds of heterogeneity with which we are presented in SPICE. Also, there are various opportunities for tuning CORBA-based systems for efficiency.

[5] Available at http://salix.riken.go.jp/sp2000_Installers/InstData/Windows/install.exe

3.2 A Flexible Architecture for SPICE

The general architecture chosen for SPICE is illustrated in Fig. 1. It is CORBA-based, as already indicated. This means that our system can be decomposed into discrete, maintainable objects; we have the flexibility to distribute objects optimally, and the implementation language independence and platform independence of CORBA ensure that we can interoperate effectively across all databases of interest. We shall now consider elements of SPICE in more detail.

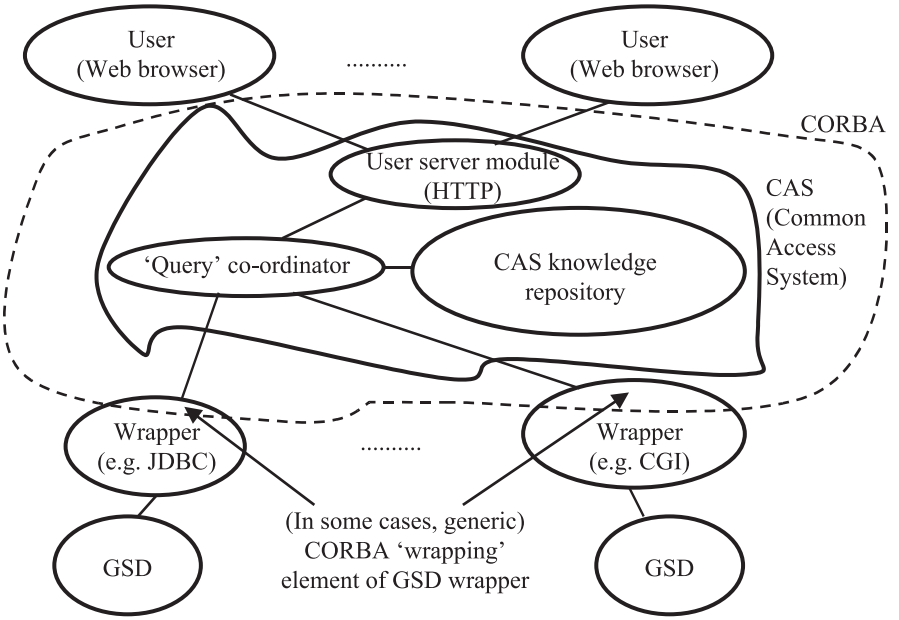


Fig. 1. The SPICE architecture

Conceptually the SPICE system can be divided into two parts:

- the *common access system* (CAS), which acts as a World-Wide Web server (brokering requests from users) and co-ordinates the formulation of suitable queries to the GSDs and the assembly of results, and
- *wrappers*, which wrap the GSDs into a common data model and materialise them on the CORBA 'bus' so that the CAS can query the GSDs.

Note that the data to be exchanged is defined using structured objects and there is no metadata used in this exchange other than what appears in the IDL. We have seen that this is appropriate since a common data model can be, and indeed has been, readily defined for SPICE, as can the set of predefined queries required.

We can therefore ensure that all wrappers wrap to this data model. Metadata is consequently redundant for communication between the CAS and the wrappers.

Now let us consider each of the elements of SPICE a little more fully. The *common access system* comprises three major components. The *user server module* acts as a World-Wide Web HTTP server, so that users running standard Internet browsers can connect to the system. This reflects a design decision that a very minimal requirement should be placed on the capabilities of the client machine. This is realised by simply requiring that it is connected to the Internet and has Internet browsing software installed. The user server module passes requests on to the *query co-ordinator*, which determines the GSDs that must be consulted, queries each of them and assembles the results. The query co-ordinator is guided by the *CAS knowledge repository*, more details of which are given in the next section. As the SPICE project progresses we will be experimenting with the contents of this repository and the ways in which they are exploited, in order to improve the knowledge used to direct (or substitute for) searches.

The *wrappers* fall into two distinct categories: 'unified' and 'divided'. The reason for this distinction is that some GSD organisations do not wish to be involved in CORBA programming directly. In a *divided* wrapper, the CAS communicates with a standard CORBA proxy wrapper which communicates in turn with the GSD using an agreed CGI protocol and retrieves responses expressed in XML, marked up in an agreed manner. It will be noted that in this kind of wrapper, any transformation between the underlying data model and the common data model reflected in the CGI and XML is the responsibility of the implementer of the CGI server. In a *unified* wrapper, in contrast, there is flexibility both in the nature of the requests sent to the GSD (although we anticipate that most GSD owners will want to connect to their databases using JDBC) and in the data model supported: the mapping takes place within the wrapper.

3.3 Challenges Within the SPICE Architecture

The architecture presented above is clearly suitable for SPICE in the sense that it 'works', and leaves us with substantial opportunities to employ techniques to achieve the performance and flexibility we require. But how can the effort associated with building wrappers be minimised? and how can we ensure good quality of service, and 'graceful degradation' when necessary?

Building Wrappers The initial wrappers have been built as one-off coding efforts. Although with experience the effort associated with building new wrappers on this basis will inevitably reduce, we want to expedite the process by:

- forming a *wrapper toolkit* of modules that can be used for solving standard problems in the development of wrappers, e.g. connecting to a database using ODBC, and returning sensible data about a species by merging in information about its subspecies if available; and

– exploring automatic wrapper generation techniques in which metadata is supplied by the programmer which will automatically lead to generation of some parts of wrappers, and the metadata can be consulted at run-time by the wrappers where appropriate.

Other research is being carried out into such aids to wrapper generation (see, e.g., [7, 14, 10, 12, 8]). But what distinguishes our work is the plurality of technologies (e.g. CGI, JDBC, semi-structured data) and of schemas that we wish to accommodate readily, and the well-defined nature of the schema into which we are mapping. In this context we aim to achieve more efficient wrapping than would be possible in the general case.

Quality of Service A user of our system may wish to receive a rapid, approximately correct response to a query, or may be willing to wait longer for a more 'correct' response. This means that the user will not *always* require that, in order to service his or her query, every single GSD must be fully consulted for the very latest changes. Species 2000 aims to create a high-quality catalogue of life, but there will inevitably be regions of the catalogue where a decision has had to be made between conflicting views that could be presented. We are seeking to provide a single, usable view of the taxonomy, with alternative taxonomies accessible via the synonymy (or possibly, later, by user selection), and we rely upon suitable experts for selection of the most appropriate taxonomic view *within* each GSD. Some parts of the catalogue will inevitably have better coverage within the available GSDs than others – we are also including 'interim checklists' to reduce this problem. Still other taxonomic groups may indeed have a corresponding GSD, but access may be relatively slow due to the wrapper technology adopted, the speed of the link or the capacity of the database itself. We have deliberately allowed such GSD limitations within SPICE because of the intellectual property rights associated with GSDs. We will not normally be supplied with GSDs; normally the agreement will be that we can *access* a GSD by an agreed means – i.e. by wrapping it. It follows that we must be able to provide a quality of service appropriate to the user's needs, and techniques must be employed to enable good performance and graceful degradation of service. We shall now briefly discuss the techniques we are developing to meet these requirements in SPICE.

CAS Knowledge Repository One of the main ways by which performance of the system can be improved is by only querying the databases that are needed on any particular occasion, or by not querying them at all – using cached information instead. The information available is as follows:

1. An *annual checklist* is to be made available centrally. This is a 'snapshot', taken at yearly intervals, of the entire catalogue. This can be used when either an individual GSD is unavailable or the user wishes deliberately to refer to this stable catalogue rather than to a continuously changing one.

2. A *taxonomic hierarchy* allows the user to narrow down the search explicitly, because if (s)he selects a particular taxonomic group then it may be that only a small number of GSDs cover that group.
3. A *genus cache* is to keep a record of which genera are found to be – and *not* to be – in each GSD.[6] This can direct vague searches towards specific GSDs without having to query the GSDs directly at this stage. This genus cache can be used to augment the on-line taxonomic hierarchy dynamically. Note also that a genus cache can be used to detect changes since the annual checklist was generated. Although in principle we could maintain a complete cache of the seven standard items of information for every species in every GSD, some potential GSD providers do not wish to allow such replication, although they are prepared to release a frozen version annually.

Change Notification The CAS knowledge repository may contain stale information: we have argued that this is a price that users will sometimes have to pay in order to obtain a rapid response. But this problem can be ameliorated if the CAS can be notified that GSDs have changed. In some cases this can be achieved through database triggers; in others, we will need to develop a suitable polling technique that the wrappers can use in order to detect updates.

4 Conclusions and Future Work

We have presented a project which is of major importance to the international biological community, emphasising the problems of interoperability that we need to solve effectively if the SPICE system is to continue to work well as the number of users and databases increases. At present we have a prototype of the CAS; unified wrappers for ILDIS and a GSD from the Natural History Museum in London (the moth family *Tineidae*), and divided wrappers for ILDIS (which can be used as an alternative to the ILDIS unified wrapper) and a GSD from the Royal Botanical Gardens, Kew in the UK (the plant order *Fagales*). We are using these to experiment with scalability by (for example) artificially splitting the GSDs into many smaller ones, or by duplicating GSDs. The techniques listed in Sect. 3.3 are being explored, with a view to generalising them so that they are applicable to other applications which, like ours, involve interoperability within a federated database having a frozen global schema in which a predefined set of queries is to be supported. Although we have sought to justify the use of CORBA as the basis of the SPICE system, we recognise the importance of existing standards, and one of our longer term aims is to make it possible for SPICE to interoperate seamlessly with systems based on standards such as Z39.50.

[6] The genus is a component of every scientific species name and identifies a small group of species

5 Acknowledgements

The research reported in this paper is based on the framework and problems presented by Species 2000, and was funded by a grant from the Bioinformatics committee of the UK BBSRC and EPSRC. We are grateful to the Species 2000 team for co-operation and access to all project details; to D. Gee, J. Shimura and Y. Ichyanagi for access to earlier CAS prototypes, and to various Species 2000 member database organisations – especially ILDIS and FishBase – for access to their data and systems.

References

[1] ANSI/NISO. *Information Retrieval: Application Service Definition and Protocol Specification*, April 1995. http://lcweb.loc.gov/z3950/agency/document.html.

[2] C. Baru et al. XML-based information mediation for digital libraries. In E. A. Fox and N. Rowe, editors, *Proceedings of the Fourth ACM Conference on Digital Libraries*, pages 214–215, N.Y., 1999. ACM Press.

[3] F.A. Bisby. Botanical strategies for compiling a global plant checklist. In F.A. Bisby, G.F. Russell, and R.J. Pankhurst, editors, *Designs for a Global Plant Species Information System*, pages 145–157. Oxford University Press, Systematics Association, 1993. Special Volume No. 48.

[4] F.A. Bisby. Putting Names to Things and Keeping Track: the Species 2000 Programme for a Coordinated Catalogue of Life. In P. Bridge, D.R. Morse, and P.R. Scott, editors, *Information Technology, Plant Pathology and Biodiversity*, pages 59–68. CAB International, Wallingford, 1997.

[5] F.A. Bisby, J.L. Zarrucchi, B.D. Schrire, Y.R. Roskov, and R.J. White. Ildis world database of legumes (edition 4). International Legume Database and Information Service, Reading, U.K., 1999. Electronic Database available by subscription (*LegumeDisc* service), and on the Web (*LegumeWeb* service) at www.ildis.org.

[6] S.M. Brandt and F.A. Bisby. Species 2000: From Planning and Prototypes to Full Implementation. *Association of Systematics Collections Newsletter*, 27:12–13, 1999.

[7] J.-R. Gruser et al. Wrapper generation for WEB accessible data sources. In *Proceedings of the 6th International Conference on Cooperative Information Systems*, pages 14–23, New York, 1998.

[8] J. Hammer et al. Template-based wrappers in the TSIMMIS system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 532–535, New York, 1997. ACM Press.

[9] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, 1990.

[10] L. Liu et al. An XML-based wrapper generator for web information extraction. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMod-99)*, pages 540–543, New York, 1999. ACM Press.

[11] D.R. Maddison and W.P. Maddison. The Tree of Life: a multi-authored, distributed Internet project containing information about phylogeny and biodiversity. http://phylogeny.arizona.edu/tree/phylogeny.html.

[12] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99)*, pages 190–197, New York, 1999. ACM Press.

[13] A. Paepcke et al. Using distributed objects for digital library interoperability. *IEEE Computer*, 29(5):61–68, 1996.

[14] M. T. Roth and P. M. Schwarz. Don't scrap it, wrap it! A wrapper architecture for legacy data sources. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pages 266–275, 1997.

[15] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[16] J. Shimura et al. Species 2000: a Java/CORBA Common Access System implemented by the Species 2000 initiative in Japan. In *Proceedings of Objects in Bioinformatics '98*, 1998. Poster presentation: abstract available at http://industry.ebi.ac.uk/˜alan/oib98/Abstracts/shimura.html.

[17] J. Siegel. OMG overview: CORBA and the OMA in enterprise computing. *Communications of the ACM*, 41(10):37–43, 1998.

[18] S. Weibel et al. Dublin core metadata for resource discovery. http://www.ietf.org/rfc/rfc2413.txt, September 1998.

[19] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

MISE: The MediaSys Image Search Engine

Frédéric Andrès[1], Nicolas Dessaigne[2,1], José Martinez[2], Noureddine Mouaddib[2], Kinji Ono[1], Douglas C. Schmidt[3], and Panrit Tosukhowong[1]

[1] NACSIS, R&D Department, Tokyo, Japan
{andres, ono}@rd.nacsis.ac.jp
[2] Polytechnic School of University of Nantes/IRIN, Nantes, France
{Jose.Martinez, Noureddine.Mouaddib}@irin.univ-nantes.fr
[3] Washington University, Saint-Louis, MI, USA
schmidt@cs.wustl.edu
panrit@net.is.uec.ac.jp

Abstract: Multimedia searching over Internet has gained substantial popularity in the past two years. Java's networking features, along with the growing number of Web browsers that can execute Java applets, facilitate distributed processing. Networking and computational performances are key concerns when considering the use of Java to develop performance-sensitive distributed multimedia search engines.
This paper makes four contributions to the study of such performance-sensitive distributed multimedia search engines. First, we describe the architecture of MISE, the **M**ediaSys **I**mage **S**earch **E**ngine over a large scale network. Secondly, we present the search capabilities of MISE as companion part of image processing. Thirdly, various evaluations of MISE have been made in terms of image processing performances and compared to the performances of Xv, an equivalent image processing application written in C.

1. INTRODUCTION AND MOTIVATION

Lot of works in the field of multimedia management systems have followed generic approaches and have developed visualisation tools in the field of search by content, mainly for images [33] [34] [25] [8] [9], and especially for the manipulation and the storage of features such as the colours, the edges, or the shape of the content of images. Also, the design of data structure has been studied for modelling [19], representing and storing multimedia content [11] as well as for accessing efficiently to multidimensional multimedia feature vectors [14] [31] [4] [17] [35]. However, problems of large scale distributed multimedia information systems such as the quality of the search, the performance, the flexibility, and the customisability have been mostly ignored. MediaSys provides such an extensible infrastructure for multimedia management. For instance, plug-ins for image operations or filters can be associated and integrated, even dynamically. This eases its upgrades according to the user's requirements and the evolving technology.
The remainder of this paper is organised as follows: Section 2 motivates and describes the architecture of our distributed multimedia system. Then, Section 3 describes more particularly the MISE component, i. e., the client part of the architecture. Section 4

M. Ibrahim, J. Küng, and N. Revell (Eds.): DEXA 2000, LNCS 1873, pp. 993–1002, 2000.

introduces MediaSys, the server. Section 5 compares the performances of MISE in terms of image processing with an equivalent image processing application written in C. Finally, we give some concluding remarks.

2. ARCHITECTURE OF THE MULTIMEDIA DISTRIBUTED SYSTEM

2.1. Motivations and General Architecture

Image manipulation plays a key role in cultural, medical, or environmental applications. The demand for systems that support image searching, visualisation, analysis and processing has increased significantly, as reported by the Gardner Group. This increase is due to the advent of the digital world and the generalisation of direct production of digital documents of any kind (text, audio, video, *etc.*). Also, due to Internet, the distribution aspect is more and more important.
Figure 1 shows the general architecture of a large-scale distributed image-based system. In this type of environment, various devices produce image data that are transferred to the multimedia storage systems, i. e., MediaSys servers. Users with various roles and profiles can search for and access to such image data using MISE client tools. They can visualise them and/or analyse them.

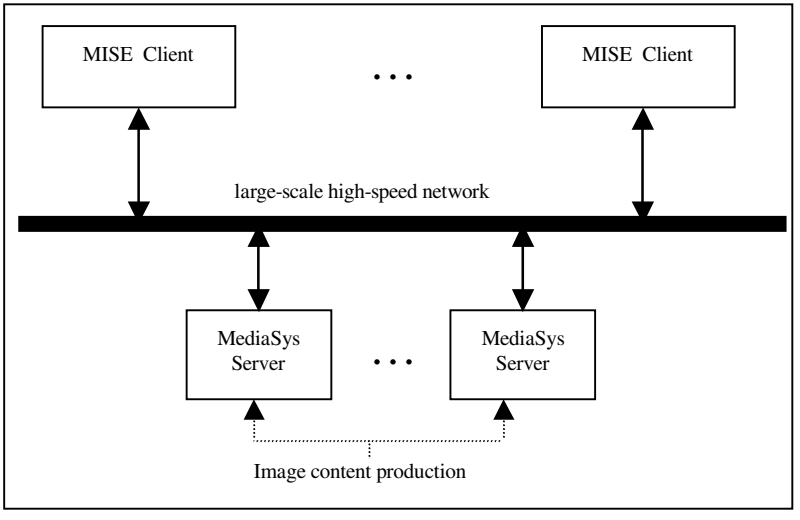


Figure 1. Architecture of the Multimedia Distributed System

The need for image search engines over distributed environment is also driven by economic factors. Actors and holders of multimedia data such as museums, hospitals, weather broadcasting companies require search engines to unify the search over their multiple and distinct image repositories. Furthermore, they are required to provide a quality of service (QoS) in terms of high performance and functionality. High-

speed networks, such as ATM or Fast Ethernet, allow the transfer of images efficiently, reliably, and economically.

2.2. Requirements of an Image Search Engine over a Large Scale Distributed Environment

A typical image search engine over a distributed environment [30] must be:

- *Extensible* to manipulate new image format and to adapt to various data models;
- *Efficient* to process image data and to deliver it to the users;
- *Scalable* to support the growing demands of searching over large-scale systems;
- *Flexible* to dynamically reconfigure image processing features to cope with changing requirements;
- *Reliable* to ensure QoS in terms of correctness and availability of image data when requested by users;
- *Cost-effective* to minimise the overhead of accessing images across large scale distributed systems;
- *Secure* to ensure that data confidentiality is preserved according to user's role.

Developing a distributed image search engine that meets all of these requirements is challenging, particularly because some features conflict with others. For instance, efficiency often requires high-performance computers and high-speed networks, thereby raising costs as the number of users increases.

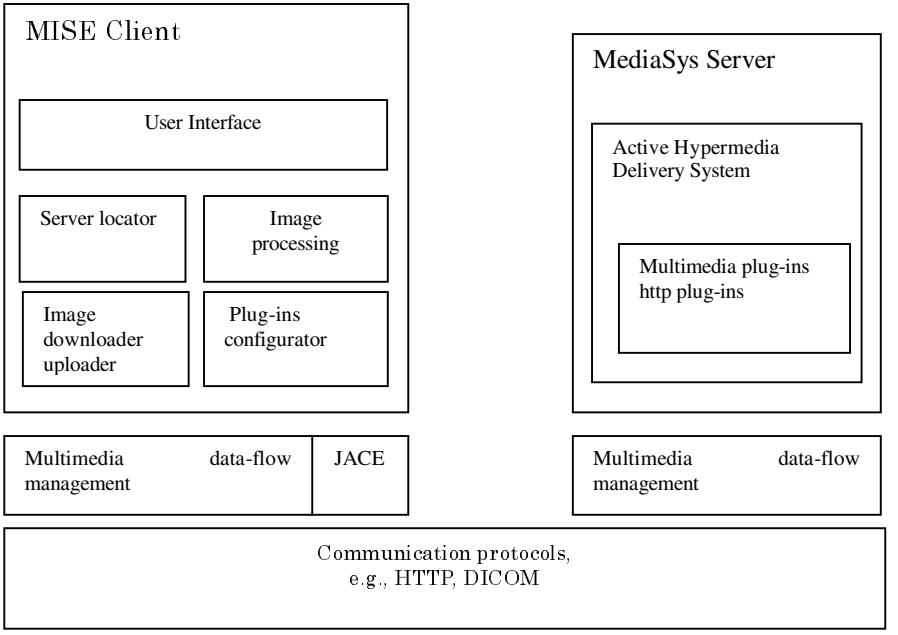


Figure 2. The MISE platform

2.3. Design of MISE platform

Figure 2 shows the detailed architecture of the MISE platform. The two major components of the initial version are the MISE client and the MediaSys server. MISE

is an evolution of Find$^{Im}_{AGE}$ developed at IRESTE/IRIN since 1996. The MediaSys server is based on the AHYDS platform (*Active HYpermedia Delivery System*) [2], which is an active hypermedia delivery system developed at NACSIS since 1995. Furthermore, the MISE tool uses the resource manager JACE [29] developed at Washington University, Saint Louis, Missouri. Finally, it also follows the Active Object approach [16]. MISE and MediaSys are discussed in the following two sections

3. THE CLIENT MISE APPLET

3.1. The Key Features of MISE

MISE allows users to search for and access to any image stored in a MediaSys server. In addition, the MISE applet provides a hierarchical browser that allows users to traverse sets of images on remote MediaSys servers. This makes it straightforward to find and select images across the network, making MISE quite usable, as well as easy to learn.

Once an image has been downloaded, image filters or image operations can process it. Although MISE is targeted for distributed image systems, it is a general-purpose image-manipulation engine. Image filters or operations can be dynamically configured into MISE via the Service Configurator pattern [28] without reloading nor restarting the MISE applet.

Conversely, a processed image can be uploaded to the image server from where the applet was downloaded thanks to the MediaSys server support.

As shown in Figure 2, there are six main components in a MISE client applet:

- *Graphical User Interface:* which provides a front-end to the image search engine and to the image-processing tool. It enables the users to search images, to receive images, to process them and to send them back to image servers (MediaSys server). Figure 3 illustrates the MISE graphical user interface (GUI.)

- *Server Locator:* which locates a URL address associated to the user role that can reference an image or an image server (MediaSys-typed server). If the URL points to a MediaSys server, its content is listed so that users can browse them to choose one image to process. This component is used by the following Image Downloader/Uploader modules.

- *Image Downloader:* which downloads an image or a set of images located by the Server Locator and displays the image or the set in the applet. The Image Downloader ensures that all pixels of the image are retrieved and displayed properly.

- *Image Uploader*: which uploads the currently displayed image to the server from where the applet was downloaded (applet security restrictions.) In addition, the MediaSys server supports data uploading using the HTTP and its PUT method. The Image Uploader writes the updated image to the server. This allows the user to save processed images persistently at the server.

- *Image Processor*: which extracts characteristics from the displayed image using either filters, or other image analysis techniques. The end of the image processing is characterised by the display of the resulting image.

- *Plug-ins Configurator* [6]: which downloads dynamically image processing plug-ins such as image filters, edge detectors, colour detectors from the Server and configures them in the applet according to the needs of the user. The Filter Configurator uses the Service Configurator pattern [29] to dynamically configure the image filters and image operation.

3.2. The MISE Search Component

The input to the search engine consists of a set of sample images along with their associated weights as given by the user. The purpose of the search engine is to retrieve an ordered list of images of decreasing similarity with respect to the user's choices. Classically, sample images with the highest weight, say 1.0, will always be retrieved (if they are in the database), whereas the ones with the lowest weight, accordingly 0.0, will never appear in the result list. A threshold has to be given in order not to retrieve about the whole database. Choosing such a threshold can be a difficult task and we rely on the default median value, i. e., 0.5, which can be tuned.

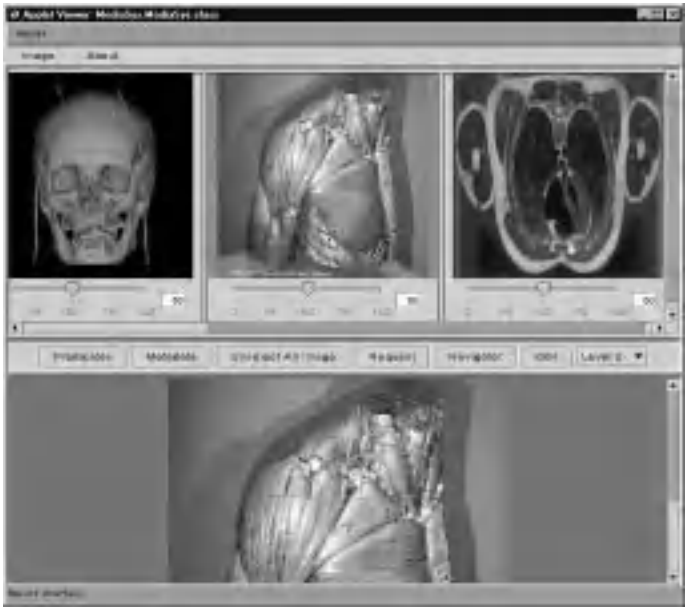


Figure 3. The MISE User Interface

3.2.1 The Relevance Feedback Loop Approach

From the user's point of view, the process has been extremely simplified. It is based on the information retrieval paradigm [12]. The end-user selects whether the currently displayed images fit his or her need or not. In Figure 3, under each image, a slider allows him or her to give a rank between 0 and 1 (default is 0.5.) The use of example and counter-example images offers advantages. First, this is easier for the end-user because it is more intuitive. Next, the graphical interaction is limited to weight annotations of *images*, rather than tuning several unintuitive knobs, as pointed out by

other authors [1] [26]. Also, it permits more expressive power than, say, a rough drawing. For instance, when drawing a picture consisting of several regions, you implicitly inform the system that all the (activated) properties of the regions are important. This is the approach of several prototypes and commercial image retrieval systems such as Picasso [10], QBIC [11], Virage [3], NeTra [18], *etc.* You cannot easily inform the system that you are interested in the size and colour of a given region, while in the shape and the texture of a second one. Trying to do that in a user-interface turns out to provide a graphical but still complex language to write down formal queries, or to accept limitations, e. g., VisualSEEK [32].
From this input, the system infers automatically one formal query that best fits the user's hints, with respect to the set of properties that have been extracted on each image and activated in the interface. This query is issued onto the database and a new list of images is displayed for another relevance feedback loop. The process continues until the user is satisfied by the returned answer, or that he or she feels that good images are not present in the database [23]. The current version of the system is based on fuzzy logic, though the underlying query language remains OQL (*Object Query Language*) [7].

3.2.2 The Indexing Process

Each image in the database has to be indexed. The indexing process consists in extracting as much features as possible. Features are regions, global histograms for colours or intensity, edges, and even meta-data such as keywords (which can be provided «automatically» through advanced image formats, e. g., the forthcoming MPEG-7.) Histograms and above all keywords provide much more precise searches thanks to semantics [26].
The segmentation process is done with respect to a given colour partitioning. The algorithm and the obtained results have been detailed in [20]. For the sake of retrieval, regions are characterised by a set of unary and binary functions, such as the horizontal position or the adjacency of two regions respectively. These functions are based on the basic and often non-intuitive features of the regions. They should provide information closer to the human visual system. In particular, each function can be based on several features, e.g., the adjacency of two regions depends on their perimeters but also on their size, and possibly on their shape [5].

3.2.3 The Query Inference

The other technical part of the querying process is to find the «best» query to issue on the database, related to the given set of sample images. This problem is NP-complete, as demonstrated in [21]. Therefore, an approximate algorithm is required, in order not to incur too high a search time. Our choice has been to use genetic algorithms [13] because among the variety of learning machine and optimisation algorithms [25] [24], they offer a good compromise between speed, «exhaustive» search, ease of paralleling in the future, but also because several of the alternative algorithms present some severe drawback with respect to the precise requirements of searching for an OQL query, e. g., «branch-and-bound» was not applicable at all due to the presence of both examples and counter-examples.
In our case, each chromosome represents a possible query. A query is a conjunction of the various functions applied to several regions that are quantified existentially. For each function, we provide an inclusion test, i. e., a disjunction of possible outcomes.

For instance, a query can be: «*Is there a region r_1 and a region r_2 where r_1 is yellow, orange, or red, and r_2 is in the centre, or the top of the image, and r_1 is adjacent to r_2*?» [22]. Each candidate query (chromosome) is applied to each sample image in order to compute its fitness, i. e., how close to the user's hint it is. The algorithm stops once the best chromosome fitness of a generation has reached a threshold value, or if a maximum number of generations have been computed.

4. PERFORMANCE EVALUATION

This section introduces the results of the performance evaluation of the MISE platform. To evaluate the performance of the MISE tool, we used the following hardware and software configuration: one MediaSys server and two MISE clients connected via Ethernet. The MISE clients run on a SUN 20 including 128 MB of RAM; the MediaSys server is running on a SUN Ultra 30, 128 MB of RAM and 9.6 GB of disks. To facilitate performance measurements, the MISE applet can be configured to run in benchmark mode. When the applet runs in benchmark mode, it computes the time (in milliseconds) required to apply filters or operations on downloaded images. The timer starts at the beginning of each image processing algorithm and stops immediately after the algorithm terminates.
We conducted the performance evaluation of the image processing capabilities: We stressed MISE (version 0.5) in terms of image processing and we compared the performance evaluation with the Xv tool (Version 3.0);

Performance Evaluation of the Image Processing Capabilities

Each tool, either MISE or Xv convert the image in 24 bits RGB images before applying any image processing. The execution time is the metric to compare the image processing.
We selected two image filters, which exhibit different computational complexities, available both in Xv and MISE. They are introduced according to their usefulness in the domain of image processing:

1. *Blur Filter*: which runs a convolution over each plane (red, green, blue) of the image, using a 3×3 convolution mask consisting of all 1/9's. Effectively, the filter replaces each pixel in the image with the mean of the pixel values in the 3×3 region surrounding the target pixel.
2. *Emboss Filter*: which applies a 3×3 convolution matrix to the image, a variation of an edge detection algorithm. Most of the image is left as a medium grey, but leading and trailing edges are turned lighter and darker grey, respectively.

Results of the Performance Evaluation
Figures 4 and 5 show image processing results from Xv and from MediaSys based on two important image filters: "Blur", and "Emboss".

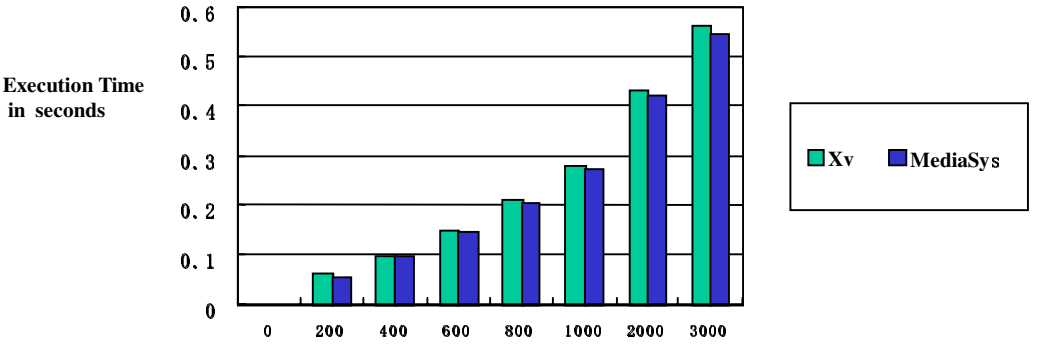


Figure 4. "Blur " filter processing

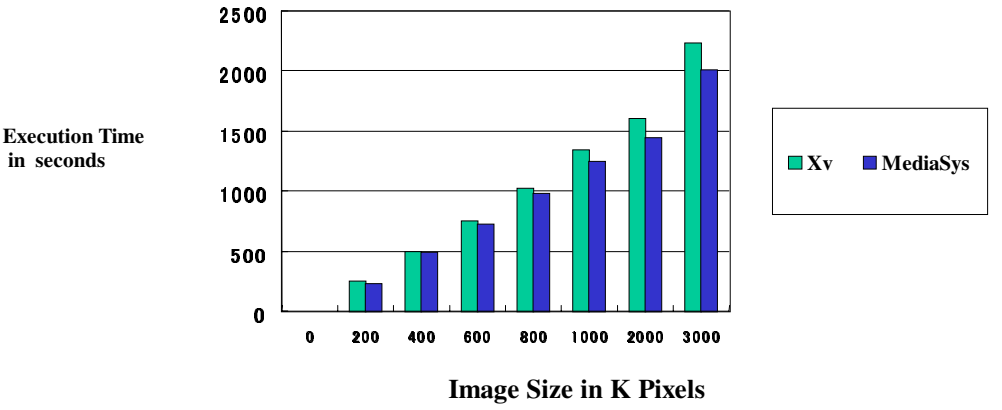


Figure 5. "Emboss" Filter processing

In any case, the image processing based on the MediaSys System is faster then Xv, the improvement is 10% for a higher reliability. Another key feature for the superiority of MISE is certainly the use of the Java Advanced Imaging library.

5. CONCLUSION

This paper described the design, the implementation, and a first performance evaluation of the MediaSys Image Search Engine (MISE), a multimedia distributed system over Internet and/or high-speed networks. This system enables the users to search, to browse, to process, and to store images according to the combination of visual and textual features with meta-data related to the images. The MediaSys servers store the meta-data, visual and textual features, and the images themselves over a large scale distributed and heterogeneous system.

We evaluated the performances of our system with respect to image processing: comparison of image filter processing speed between our system and Xv pointed out

that MediaSys processes slightly better, approximately 10 %. As conclusion, MISE, the first client tool of the MediaSys project, gains its efficiency by the use of Java. The Java approach enables to build valuable tool as MISE is a simple, portable, and distributed system.

6. REFERENCES

1. Ahanger, G., Little, T. D. C.: A Survey of Technologies for Parsing and Indexing Digital Video; Journal of Visual Communication and Image Representation, (Special Issue on Digital Libraries) (1996), Vol. 7, No. 1, pp. 28-43
2. Andrès, F. , Ono, K.: Active Hypermedia Delivery System; Proc. of the Int'l Conf. On Data Engineering (ICDE'98), Florida, (1998), pp. 600.
3. Bach, J. R., Fuller, C., Gupta, A., Hampapur, A., Horowitz, B., Humphrey, R., Jain, R. C., Shu, C.: Virage Image Search Engine: An Open Framework for Image Management; Proc. of the IS&T/SPIE Int. Symposium on Electronic Imaging: Science and Technology, Storage & Retrieval for Image and Video Databases IV (EI'96), (1996), pp. 76-87
4. Beckmann, N., Kriegel, H. P., Schneider, R., Seeger, B.: The R*-Tree: an Efficient and Robust Access Method for Points and Rectangles; Proc. of ACM SIG Int'l Conf. on Management of Data (SIGMOD'90), Atlantic City, NJ,(1990), pp. 322-331
5. Bloch, I., Maître, H.: Extending Adjacency to Fuzzy Sets for Coping with Imprecise Image Objects; Proc. of the 9th Int'l Conf. on Image Analysis and Processing (ICIAP'97), Vol. II, Florence, Italy, (1997), pp. 30-37 (in LNCS No. 1311)
6. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-oriented Software Architecture – A System of Patterns; Wiley and Sons, (1996).
7. Cattel, R. G. G., Barry, D., Bartels, D., Berler, M., Eastman, J., Gamerman, S., Jordan, D., Springer, A., Strickland, H., Wade, D.: The Object Database Standard: ODMG 2.*0*; Morgan Kaufmann Publishers, Inc., San Francisco, California, (1997), 270 p.
8. Chang, J.-W., Srivastava, J.: Spatial Match Retrieval Using Signature Files for Iconic Image Databases; Proc. of IEEE Multimedia Conference, Ottawa, Canada, (1997), pp. 658-659
9. Chang, S., Smith, J., Beigi, M., Benitez, A.: Visual Information Retrieval from Large Distribued Online Repositories; Communications of the ACM, Vol. 40, No. 12, (1997), pp. 63-71
10. Corridoni, J. M., Del Bimbo, A., Vicario, E.: Image Retrieval by Color Semantics with Incomplete Knowledge; Journal of the American Society for Information Science, Vol. 49, No. 3, (1998), pp. 267-282
11. Flickner, M., Sawhnery, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., Yanker, P.: Query by Image and Video Content: The QBIC System; IEEE Computer, (1995), pp. 23-32
12. Frakes, W. B., Baeza-Yates, R.: Information Retrieval: Data Structures & Algorithms; Prentice-Hall, (1992), 504 p.
13. Goldberg, D. E.: Genetic Algorithms; Addison-Wesley, (1991)
14. Guttman, A.:R-Trees: A Dynamic Index for Geometric Data; Proc. of ACM SIG Int'l Conf. on Management of Data (SIGMOD'84), (1984), pp. 47-57
15. Jain, P., Schmidt, D. C.: Experiences Converting a C Communication Software Framework to Java; C Report, Vol. 9, (1997)
16. Lavender, R. G., Schmidt, D. C.: Active Object: an Object Behavioral Pattern for Concurrent Programming; in Pattern Languages of Program Design, Reading, MA : Addison-Wesley, (1996).
17. Lin, K. I., Jagadish H. V., Faloutsos, C.: The TV-Tree An Index Structure for High Dimentional Data; The VLDB Journal, 3(4), (1994), pp. 517-542

18. Ma, W.Y., Manjunath, B. S.;:*NETRA: A toolbox for navigating large image databases*; Proc. of the IEEE Int'l Conf. on Image Processing (ICIP'97), 1997
19. Martinez, J.: The Design of an Extensible Multimedia Library for an OODBMS; Proc. of the 7th IEEE Int'l Workshop on Database and Expert Systems Applications (DEXA'96), Zurich, Switzerland, 5-7 (1996), pp. 208-213
20. Martinez, J., Guillaume, S.: Colour Image Retrieval Fitted to Classical Querying; Proc. of the 9th Int'l Conf. on Image Analysis and Processing (ICIAP'97), Vol. II, Florence, Italy, (1997), pp. 14-21 (in LNCS No. 1311)
21. Martinez, J., Guillaume, S.: Colour Image Retrieval Fitted to Classical Querying; Networking and Information Systems Journal, Vol. 1, No 2-3, (1998), pp. 251-278 (Editions HERMES, Paris, ISBN 2-86601-731-5)
22. Martinez, J., Marchand, S.: Towards Intelligent Retrieval in Image Databases; Proc. of the 5th Int'l Worshop on Multi-Media Data Base Systems (MMDBMS'98), August 5-7, (1998), Dayton, Ohio, USA, pp. 38-45 (IEEE Computer Press, Los Alamitos, California, USA, ISBN 0-8186-8676-6)
23. Martinez, J., Mouaddib, N.;Multimedia and Databases: A Survey; Networking and Information Systems Journal (NISJ), Vol. 1, No. 6, (1999) (Editions HERMES, Paris)
24. Michalski, R. S., Bratko, I., Kubat, M. (Eds.): Machine Learning and Data Mining: Methods and Applications; John Wiley & Sons, New York, (1998), 456 p.
25. Mitchell, T. M.: Machine Learning; WCB/MacGraw-Hill, (1997), 414 p.
26. Ogle, V. E., Stonebraker, M.; *Chabot: Retrieval from a Relational Database of Images*; IEEE Computer, (1995), pp. 40-48
27. Santini, S., Jain, R.: Beyond Query by Example; Proc. of the 6th Int'l ACM Conf. on Multimedia (MM'98), Bristol, UK, (1998)
28. Schmidt, D. C.: Acceptator and Connector: Design Patterns for Initializing Communication Services; Pattern Languages of Program Design (Martin, R., Buschmann, F., Riehle, D., eds.), reading, MA, Addison-Wesley, 1997.
29. Schmidt, D. C., Suda, T.: An Object-oriented Framework for Dynamically Configuring Extensible Distributed Communication Systems; IEE/BCS Distributed Systems Engineering Journal (Special Issue on Configurable Distributed Systems), Vol.2, December 1994, pp. 280-293
30. Schmidt, D. C., Harrison, T. H., Al-Shaer, E.; *Object-Oriented Components for High-speed Network Programming*; Proc. of the 1st Conf. on Object-Oriented Technologies and Systems, Monterey, CA, USENIX, (1995)
31. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R+-Tree: A Dynamic Index of Multidimensional Objects; Proc. of the 18th Int'l Conf. on Very Large Data Bases (VLDB'87), (1987)
32. Smith, J. R., Chang, S.-F.: VisualSEEK: a Fully Automated Content-Based Image Query System; ACM Multimedia, Boston, Massachussets, (1996)
33. Swain, M. J., Ballard, D. H.: Color Indexing; International Journal of Computer Vision, 7(1), (1991)
34. Swain M. J.: Interactive Indexing into Image Databases; Proc. of SPIE, Storage and Retrieval for Image and Video Databases, Vol. 1908, San Jose, California, (1993), pp. 95-103
35. Wang, W., Yang, J., Muntz, R.: STING: A Statistical Information Grid Approach to Spatial Data Mining; Proc. of the 23rd Int'l Conf. on Very Large Data Bases (VLDB'97), Athens, Greece, (1997), pp. 186-195

Author Index