Lecture Notes in Computer Science 1552

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

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

Yahiko Kambayashi Dik Lun Lee
Ee-Peng Lim Mukesh Kumar Mohania
Yoshifumi Masunga (Eds.)

Advances in Database Technologies

ER '98 Workshops on
Data Warehousing and Data Mining,
Mobile Data Access, and
Collaborative Work Support and
Spatio-Temporal Data Management
Singapore, November 19-20, 1998
Proceedings

Springer

Volume Editors

Yahiko Kambayashi
Kyoto University, Graduate School of Informatics, Dept. of Social Informatics
Yoshida Sakyo Kyoto 606-8501, Japan, E-mail: yahiko@isse.kuis.kyoto-u.ac.jp

Dik Lun Lee
Hong Kong University of Science and Technology, Dept. of Computer Science
Clear Water Bay, Hong Kong, China, E-mail: dlee@cs.ust.hk

Ee-Peng Lim
Nanyang Technological University, School of Applied Science
N4-2A-12, Nanyang Avenue, Singapore 639798, E-mail: aseplim@ntu.edu.sg

Mukesh Kumar Mohania
University of South Australia, School of Computer and Information Science
The Levels Campus, Mawson Lakes, 5095, S.A., Australia
E-mail: mohania@cs.unisa.edu.au

Yoshifumi Masunaga
Ochanomizu University, Faculty of Science, Dept. of Information Sciences
2-1-1 Otsuka, Bunkyo-ku, 112-8610 Tokyo, Japan
E-mail: masunaga@is.ocha.ac.jp

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Advances in database technologies : proceedings / ER '98 ; Workshops on Data Warehousing and Data Mining, Mobile Data Access, and Collaborative Work Support and Spatio Temporal Data Management, Singapore, November 19 - 20, 1998. Yahiko Kambayashi ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1999
(Lecture notes in computer science ; Vol. 1552)
ISBN 3-540-65690-1

CR Subject Classification (1998): H.2, H.3-5, C.2.4-5

ISSN 0302-9743
ISBN 3-540-65690-1 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10693180 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Foreword

Advances in Database Technologies (ADT) is a workshop series designed to promote interaction and discussion among researchers and practitioners in specific topics related to advanced data modeling and database technologies. The first ADT workshop was held in conjunction with the 17th International Conference on Conceptual Modeling (ER'98) in Singapore. It consisted of three small workshops, i.e., the "International Workshop on Data Warehousing and Data Mining" organized by Prof. Sham Navathe and Dr. Mukesh Mohania, the "International Workshop on Mobile Data Access" organized by Prof. Dik Lun Lee, and the "International Workshop on New Database Technologies for Collaborative Work Support and Spatio-Temporal Data Management" organized by Prof. Yoshifumi Masunaga. These three small workshops were held simultaneously on 19 and 20 November, and the papers were jointly published in this ADT'98 proceedings.

We would like to thank Prof. Sham Navathe, Dr. Mukesh Mohania, Prof. Dik Lun Lee, Prof. Yoshifumi Masunaga and their program committees for working together to insure an excellent workshop program. The program co-chair of ER'98 Prof. Tok Wang Ling and his organizing committee members gave outstanding support to the workshops with the local arrangements, registration, publicity, and the preparation of the proceedings. Our special thanks go to Prof. Stefano Spaccapietra, Chair of the ER Steering Committee, for his encouragement to make the ADT workshop an annual event to be held in conjunction with the future ER conferences. We would also like to thank Dr. Peter Chen and Dr. Bernhard Thalheim for their efforts in expanding the scope of the ER conference series. We were fortunate to have ACM, ACM SIGMOBILE, the ER Insitute and the Japanese research project on Advanced Databases (supported by the Ministry of Education, Science, Sports and Culture of Japan) as workshop sponsors. Lastly, we would also like to express our sincere thanks to all reviewers of the workshop papers.

We hope that the first ADT workshop was stimulating and enjoyable to all who attended it, and we look forward to more successful ADT workshops in the coming years.

December 1998

Prof. Yahiko Kambayashi
ER'98 Conference Chair

Dr. Ee-Peng Lim
ER'98 Workshop Chair

Workshop Chairs' Messages

Data warehousing and data mining is a recent information technology that allows electronic information to be easily and efficiently accessed for decision making activities and the automated extraction of interesting patterns and trends in the data stored in a data warehouse (called data mining).

The first international workshop on Data Warehousing and Data Mining (DWDM'98) focuses on the physical and the logical design of data warehousing and data mining systems. The scope of the papers covers the most recent and relevant topics in the areas of data and web warehousing, multidimensional databases, knowledge discovery and data mining.

We received more than 30 papers from over 15 countries and the programme committee finally selected 14 papers. A special panel on 'Data Warehousing and Data Mining - Are we working on the right things?' is included in the workshop programme to discuss the current challenges.

We would like to thank the conference general chair, workshop general chair and the organising committee of the 17th International Conference on Conceptual Modelling (ER'98) for recognising the importance of the theme of this workshop. We are very indebted to all programme committee members and the referees who have very carefully and timely reviewed the papers. We would also like to thanks all the authors who submitted their papers to this workshop.

November 1998

Sham Navathe
Workshop Chair

Mukesh Mohania
Program Committee Chair

International Workshop on
Data Warehousing and Data Mining

Wireless communication has added a new dimension of freedom to the Internet. It gives people truly ubiquitous access to information and computational resources, practically at any time and from anywhere in the world. In the past few years, we have witnessed the astonishing progress in both research and technological advancement in wireless communication systems. We expect to see a rapid penetration of the technology into every aspect of life, from global communication to applicance control in households. This workshop was organized with this perspective in mind to provide a forum for this timely topic. To align with the theme of the main conference (Conceptual Modelling, ER '98), this workshop was intended to focus on the data access aspect of mobile systems. I am glad that with the help of the program committee, we were able to put together an interesting program drawing papers from industry and academia around the world, with a panel on the future directions of academic research and industrial development. I am also greatly indebted to Dr. Ee-Peng Lim for his tremendous efforts in handling all the organizational aspect of the workshop and to Dr. Tok Wang Ling for making the publication of the workshop proceedings possible.

I hope this workshop is merely the very first of a continuous sequence of workshops dedicated to the data access aspect of mobile computing.

December 1998

Dik L. Lee
Chair, International Workshop on
Mobile Data Access

The International Workshop on New Database Technologies for Collaborative Work Support and Spatio-Temporal Data Management (NewDB'98) was held on November 19-20, 1998 in Singapore. The NewDB'98 was organized in conjunction with the Seventeenth International Conference on Conceptual Modeling (ER'98). These proceedings contain the technical papers accepted for presentation at the workshop.

The NewDB'98 program committee accepted 12 and 6 papers for regular and short presentation, respectively, which were selected from 28 submitted papers from various countries. It is interesting to note that the majority of the 13 accepted papers were from Japan. In Japan there is an on-going project "Research and Development of Advanced Database Systems for Integration of Media and User Environments" funded by the Ministry of Education, Science, Sports and Culture. This is a three year project started in 1996, and is directed by Prof. Yahiko Kambayashi of Kyoto University. In carrying out the project, we have recognized that the support of collaborative work and management of spatio-temporal data has become one of the most interesting and important database applications, which is due to the tremendous progress of database and its surrounding technologies in the last decade.

I believe that the future database systems in the coming twenty first century cannot survive without supporting spatio-temporal virtual database environments which users enter to work together.

Last but no least, I would like to thank you all of the NewDB'98 program committee members, Dr. Ee Peng Lim of ER'98 Workshop Chair, and Prof. Tok Wang Ling of ER'98 Program Co-Chair for their hard work.

December 1998

Yoshifumi Masunaga
Chair, International Workshop on
New Database Technologies for
Collaborative Work Support and
Spatio-Temporal Data Management

Workshop Organization

International Workshop on Data Warehousing and Data Mining
Chair: Sham Navathe (Georgia Institute of Technology, USA)
Program Chair: Mukesh Mohania (University of South Australia, Australia)

International Workshop on Mobile Data Access
Chair: Dik L. Lee (Hong Kong University of Science and Technology, Hong Kong)

International Workshop on New Database Technologies for Collaborative Work Support and Spatio-Temporal Data Management
Chair: Yoshifumi Masunaga (University of Library and Information Science, Japan)

Conference Chair, ER'98:
Yahiko Kambayashi (Kyoto University, Japan)

Program Co-Chairs, ER'98:
Tok Wang Ling (National University of Singapore, Singapore)
Sudha Ram (University of Arizona, USA)

Workshop Chair, ER'98:
Ee-Peng Lim (Nanyang Technological University, Singapore)

Publication Chair:
Chuan Heng Ang (National University of Singapore, Singapore)

Registration Chair:
Hock Chuan Chan (National University of Singapore, Singapore)

Finance Chair:
Cheng Hian Goh (National University of Singapore, Singapore)

Local Arrangements Co-Chairs:
Mong Li Lee (National University of Singapore, Singapore)
Danny Poo (National University of Singapore, Singapore)

Industrial Chair:
Kian Lee Tan (National University of Singapore, Singapore)

Publicity Chair:
Yong Meng Teo (National University of Singapore, Singapore)

Secretary:
Sew Kiok Toh (National University of Singapore, Singapore)

Steering Committee Representatives:
Stefano Spaccapietra (Swiss Federal Institute of Technology, Switzerland)
Bernhard Thalheim (Cottbus Technical University, Germany)
Peter Chen (Louisiana State University, USA)

Regional Co-ordinators:
Alberto Laender (Federal University of Minas Geras, Brazil)
Erich Neuhold (German National Research Center for Information Technology, Germany)
Masaaki Tsubaki (Data Research Institute, Japan)
Shiwei Tang (Peking University, China)

Program Committee for International Workshop on Data Warehousing and Data Mining

Sharma Chakravarthy, USA
Arbee L.P. Chen, Taiwan
Qiming Chen, USA
Anindya Datta, USA
Jonathan Gray, Australia
John Harrison, Australia
Yahiko Kambayashi, Japan
Kamal Karlapalem, Hong Kong, China
Masaru Kitsuregawa, Japan
Shin'ichi Konomi, Germany
Vijay Kumar, USA
Wolfgang Lehner, Germany
Leonid Libkin, USA
Sanjay Madria, Singapore
Wee-Keong Ng, Singapore
Maria Orlowska, Australia
Stefano Paraboschi, Italy
D. Janaki Ram, India
John Roddick, Australia
N. L. Sarda, India
Ashok Savasere, USA
Ernest Teniente, Spain

Program Committee for International Workshop on Mobile Data Access

Arbee Chen, Taiwan
Wan-Sup Cho, Korea
Pam Drew, USA
San-Yih Hwang, Taiwan
Wang-Chien Lee, USA
Bo Li, Hong Kong
Ee-Peng Lim, Singapore
Sanjay Kumar Madria, Singapore
Kimmo Raatikainen, Finland
Nobuo Saito, Japan
Kian-Lee Tan, Singapore
Martti Tienari, Finland
Guoren Wang, China
Lawrence Yeung, Hong Kong

Program Committee for International Workshop on New Database Technologies for Collaborative Work Support and Spatio-Temporal Data Management

Masatoshi Arikawa, Japan
Hiroshi Arisawa, Japan
Chin-Chen Chang, Taiwan
David Wai-lok Cheung, HK
Max Egenhofer, USA
Ramez Elmasri, USA
Burkhard Freitag, Germany
Theo Haerder, Germany
Yoshinari Kanamori, Japan
Kyuchul Lee, Korea
Wen-Syan Li, USA
Tok Wang Ling, Singapore
Akifumi Makinouchi, Japan
Songchun Moon, Korea
Yasuaki Nakamura, Japan
Anne Ngu, Australia
Shogo Nishida, Japan
Yutaka Ohsawa, Japan
Christine Parent, Switzerland
N. L. Sarda, India
Sigeru Shimada, Japan
Mark Sondheim, Canada
Katsumi Tanaka, Japan
Seiichi Uchinami, Japan
Shunsuke Uemura, Japan
Yasushi Yamaguchi, Japan

Organized By

School of Computing, National University of Singapore

Sponsored By

ACM

ACM SIGMOBILE

Research and Development of Advanced Database Systems for Integration of Media and User Environments
(Supported by the Ministry of Education, Science, Sports and Culture, Japan)

The ER Institute

In-cooperation with

School of Applied Science, Nanyang Technological University

Singapore Computer Society

Information Processing Society of Japan

Table of Contents

1.4: Multidimensional Databases

Session chair: Mukesh Mohania, AUSTRALIA

Panel Discussion:

1.5: Data Warehouse Design Issues

Session chair: Sanjay K. Madria, SINGAPORE

Part 2: Mobile Data Access

2.1: Caching

Session chair: Sanjay K. Madria, SINGAPORE

2.2: Data Dissemination

Session chair: Hiroki Takakura, JAPAN

2.3: Replication

Session chair: Kam-yiu Lam, HONG KONG

2.4: Mobile Networks

Session chair: Stuart Jacobs, USA

2.5: Mobile Platforms

Session chair: Keizo Saisho, JAPAN

2.6: Tracking and Monitoring

Session chair: San-Yih Hwang, TAIWAN

Panel Discussion:

Part 3: New Database Technologies for Collaborative Work Support and Spatio-Temporal Data Management

3.1: Collaborative Work Support-1

Session chair: Hiroshi Arisawa, JAPAN

3.2: Collaborative Work Support-2

Session chair: Lutz Wegner, GERMANY

3.3: Temporal Data Modelling

Session chair: Stefano Spaccapietra, SWITZERLAND

3.4: Moving Objects and Spatial Indexing

Session chair: Yutaba Ohsawa, JAPAN

3.5: Spatio-Temporal Databases

Session chair: Shogo Nishida, JAPAN

3.6: Video Database Content

Session chair: Yoshinari Kanamori, JAPAN

A Fuzzy Attribute-Oriented Induction Method for Knowledge Discovery in Relational Databases

Noureddine Mouaddib and Guillaume Raschia

IRIN - Université de Nantes
2 rue de la Houssinière, BP 92208, 44322 Nantes Cedex 3, France
{mouaddib,raschia}@irin.univ-nantes.fr

Abstract. In this paper, we propose a fuzzy attribute-oriented induction method for knowledge discovery in relational databases. This method is adapted from the DBLearn system by representing background knowledge with fuzzy thesauri and fuzzy labels. These models allow to take into account inherent imprecision and uncertainty of the domain representation. We also show the power of fuzzy thesauri and linguistic variables to describe gradations in the generalization process and to handle exceptions.

1 Introduction

The computerization of business activities produces an amount of data recorded in databases. These databases are rapidly growing and managers would like to explore this mine of information. This need leads up to researches on Knowledge Discovery in Databases (KDD), considered as *the non trivial process of identifying valid, novel, potentially usefull, and ultimately understandable pattern in data* [1].

Our approach addresses this issue: it is based on the principle of attribute-oriented induction proposed in the DBLearn system [2,3] which integrates the learning-from-example paradigm, with relational database operations. DBLearn first collects the interesting data using a SQL query. Next, it applies the generalization operator attribute by attribute on the extracted relation. Some attribute values in the relation are replaced with higher level concepts defined into a partially ordered hierarchy - general-to-specific concepts tree. On the final step, the generalized relation is transformed into rules.

Our paper is motivated by two observations. First, the user domain knowledge representation is usually vague, incomplete or uncertain, even if the user is an expert. Second, as concept trees represent preferences and points of view of the user, they are necessary to provide powerful and flexible tools to represent the knowledge about the domain. As you can see, our goal is not to solve the inherent problems of induction process, but only to take advantage of an establishment in order to improve efficiency and accuracy of attribute-oriented induction methods, by using the DBLearn algorithm as a general framework.

The remainder of this paper is organized as follows. In section 2, we show how fuzzy logic offers a framework to represent inherent uncertainty or vagueness of

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 1–13, 1999.

domain knowledge. We present the advantage of using fuzzy thesauri and fuzzy labels instead of concepts trees as it is used in DBLearn. Section 3 proposes the fuzzy attribute-oriented induction method, which adapts and improves the DBLearn algorithm. We conclude in section 4 and consider future works.

2 Background Knowledge Representation

In this paper, we propose two kinds of models which are able to represent complex (uncertain, incomplete...) information. The first one (cf section 2.1) represents symbolic values in a fuzzy thesaurus [4]. The second one represents numeric values.

2.1 Representing Symbolic Values in a Fuzzy Thesaurus

General definition : A fuzzy thesaurus is a set (C, R, L, I) where :

• C is a set of concepts (or terms) $\{c_1, \ldots, c_n\}$; e.g. $\{Boa,\ Oviparous,\ Reptile,\ Snake,\ Viper\}$.

• R is a set of relations $\{r_1, \ldots, r_m\}$; e.g. $\{isa,\ isn\}$ is a set of relations where *isa* means "is a" and *isn* means "is not a".

• I is a given set of degrees in $[0, 1]$.

• L is a partial function from $C \times R \times C$ into I. The function L generates a set of links $\{l_1, \ldots, l_p\}$ where l_k represents the equality $L(c_i, r_q, c_j) = \alpha$ noted :

$$l_k : c_i \xrightarrow{r_q}_{\alpha} c_j$$

where $c_i, c_j \in C$, $r_q \in R$ and $\alpha \in I$ (used to express the strength of link l_k). We note $origin(l_k) = c_i$ the origin of l_k, $end(l_k) = c_j$ the destination of l_k, $rel(l_k) = r_q$ the relation involved in l_k, and $strength(l_k) = \alpha$ the degree of the link. For example (cf figure 1), $Viper \xrightarrow{isa}_{1} Snake$ is a link.

Paths A path from a concept c_i to a concept c_j, noted $path(c_i, c_j)$ is a set of links $\{l_{k_1}, \ldots, l_{k_q}\}$ such that :

$l_{k_1}, l_{k_2}, \cdots, l_{k_q} \in L$, $origin(l_{k_1}) = c_i$, $end(l_{k_q}) = c_j$, $end(l_{k_m}) = origin(l_{k_{m+1}})$ for $m = 1, \cdots, q-1$

Valid paths Some paths are not valid, for instance, "the neighbour of my neighbour is not my own neighbour". Thus, we use a grammar to define the set of valid paths.

We consider G as a Chomsky's grammar generating a set of sentences. A sentence represents an authorized sequence of relations in a path with respect to grammar G. We note this set $\mathcal{L}(G)$.

A valid path from a concept c_i to a concept c_j respecting a grammar G, noted G–$path(c_i, c_j)$, is a path $\{l_{k_1}, \ldots, l_{k_q}\}$ such that the sequence $rel(l_{k_1}), \ldots, rel(l_{k_q})$ is in $\mathcal{L}(G)$.

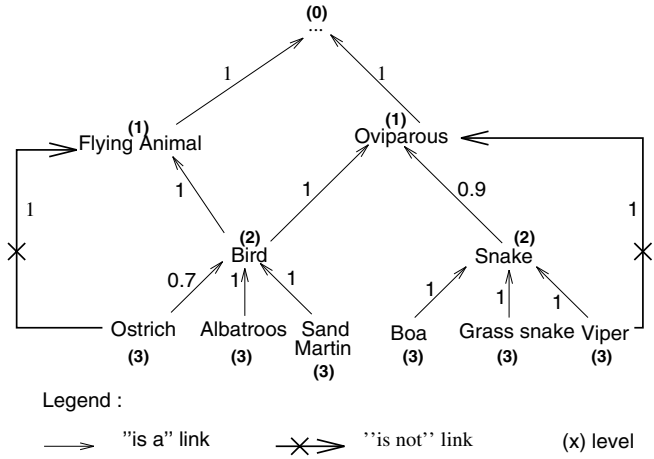


Fig. 1. An extract of the thesaurus on the attribute "type": thes1

Generalization grammar In this paper, we focus on a particular grammar used in the knowledge discovery process: generalization grammar. We consider two kinds of relations: *isa* (is a) and *isn* (is not a) like those represented in the example of figure 1. Then, to give the set of more general concepts of a concept c_i, we use a grammar defined by:

$G : X \longrightarrow isa^{+}(1) \mid isa^{\star}\ isn\ isa^{\star}(0)$

where the number between parenthesis indicates the authorization of the generated sentence (i.e. 0 for an unauthorized path and 1 for an authorized path). From the figure 1, *Albatross* can be generalized to *Bird*, *Flying Animal* and *Oviparous*. However, *Ostrich* can be generalized to *Bird* and *Oviparous* but not to *Flying Animal* because there exists an invalid path $\{Ostrich \xrightarrow{\text{isn}}_1 Flying\ Animal\}$.

Generalized concept degree A degree d is associated to each higher level concept c_j to show to which extent it generalizes the concept c_i. This degree is calculated by:

$$
\begin{aligned}
d &= \alpha \times \gamma \\
\alpha &= \max_{k=1,\cdots,m} (strength_{path}(G\text{–}path_k(c_i, c_j))) \\
\gamma &= \min_{k=1,\cdots,m} (\max(\beta(G\text{–}path_k(c_i, c_j)), \\
&\qquad 1 - strength_{path}(G\text{–}path_k(c_i, c_j))))
\end{aligned}
$$

where:

• $strength_{path}(p)$ gives the strength associated to path p (i.e. a kind of proximity degree between the first and the last term of the path). The strength of a path $p = \{l_{k_1}, \ldots, l_{k_q}\}$ is calculated by:

$$strength_{path}(p) = Min_{p=1,\ldots,q}(strength(l_{k_p}))$$

- $\beta(p)$ gives the authorization of a path p
- α is the degree of the best path from c_i to c_j (with the higher strength)
- γ is an aggregation of authorized level for all paths from c_i to c_j

From the previous example, *Ostrich* can be generalized to *Bird* with the degree 0.7, to *Oviparous* with the degree 0.7, and to *Flying Animal* with the degree 0. Then, we consider the set of more general concepts of a given concept c_i as a fuzzy set determinated by a membership function noted μ_{c_i} where $\mu_{c_i}(c_{i'}) = d$ is the generalization degree of c_i to $c_{i'}$. For example, we have $\mu_{Ostrich}(Oviparous) = 0.7$ from figure 1.

Using fuzzy thesauri, instead of concepts trees as in DBLearn allows to represent graduations of the generalization process. For example, we can represent in the thesaurus (cf figure 1) that a concept *Ostrich* can be generalized to the concept *Bird* with a weight lower than the concept *Albatroos*. If we consider the concept *Bird* as the union of the following characteristics: to fly, to have a beak, to have feathers..., an ostrich does not verify all these properties.

Exceptions are also easily taken into account with the "is not" link of fuzzy thesauri. Unlike, to represent exceptions in a hierarchy, artificial classes are introduced, for example "no flying bird", "no feathers bird"...

As the fuzzy thesaurus should be a graph, multi-generalization is allowed representing multiple points of view in the same hierarchy.

2.2 Representing Numeric Values with Fuzzy Labels

We use fuzzy labels [5] to generalize numerical values. They partition the attribute domain into fuzzy subsets. For example, the domain of the attribute "height" can be divided into three fuzzy labels "small", "medium", "tall".

In [6], a similar method is used to reduce a relation, but authors did not use the membership grade during the data reduction process to weight the generalized tuple obtained after substitution.

Compared to other approaches in KDD, fuzzy labels offer more flexibility to represent the background knowledge, particularly when dealing with values at the limit of a class. For example, the tuple $<$ *viper, 0.6 cm, ...* $>$ can be generalized in $<$ *viper, small (0.8), ...* $>$ and in $<$ *viper, medium (0.2), ...* $>$.

3 Fuzzy Attribute-Oriented Induction Method

The induction process is the most important part of works in machine learning. Its general principle consists in identifying symbolic concepts from observations [12]. Therefore this activity is really close from generalization process of data. Symbolic concept is characterized by an intentional description of the generated categories - concepts, or classes -, as opposed to the clustering activity [8], which only provides an extentional description of categories: the disjunction of all the instances of the class. In our point of view, summarized descriptions are built from background knowledge - fuzzy thesauri and fuzzy labels - and acquired

concepts are in fact merged tuples. We could notice that the word *concept* is used both for a merged tuple and for a term of a thesaurus.

Overall, the induction method we want to use is monothetic; it only considers one property at a time for the abstraction process, as opposed to a polythetic approach, which could take into account a lot of different attributes in order to define the best data generalization. So we could roughly compare our approach with the construction of decision trees [9] in classification problems.

Finally, the learning process we propose has several features in common with constructive conceptual clustering algorithms [11] :

- the unsupervised nature of the learning task : the system must cluster and merge instances without any advice or help from a human expert;
- the nonincremental process determined by the need of data global view for the choice of future-merged tuples, but we could quite easily consider an extension of the system which will integrate an incremental process in order to deal with large and evolutive databases;
- the use of background knowledge, which allows a high level of abstraction and a relevant semantics of concept descriptions;
- the great interpretability of the acquired concepts due to the use of the same vocabulary between the domain expert and the system;

Nevertheless, the main feature of a typical conceptual clustering algorithm is the top-down construction of a concept hierarchy [10], and consequentely the definition of different abstraction levels. In our approach, there does not exist this kind of data organization, even if we could take into account different abstraction levels in the generalization process.

To apply the attribute-oriented induction, we have adapted the steps proposed in DBLearn system to use fuzzy thesauri and fuzzy labels during the generalization process. In a first step, DBLearn collects relevant data from the database using a SQL query. The second step removes attributes which have no significance in the process. In the third one, the procedure selects the attribute on which the generalization operator will be applied (step four). In step five, identical tuples are merged into an unique one. These last three steps are repeated until the number of tuples in the generalized relation is greater than a threshold set by the user.

The procedure that performs the fuzzy attribute-oriented induction method is noted:

`FAOI(DB, Lat, FT, FL, thres): GR`

where :

`DB` is the database that contains data on which the algorithm will be performed.

`Lat` is a list that indicates for each attribute the number of distinct values an attribute may have in the final relation `GR`. If a thesaurus is associated to this attribute, `Lat` indicates the maximum steps in the thesaurus the generalization process may climb. For example, if the user indicates a level 2 for the attribute "type" in the relation $\mathcal{R}_{animal}$ (cf table 1), the generalization process will actually climb until the values *Bird* or *Snake* (cf figure 1).

FT and FL which are respectively the set of thesauri and the set of fuzzy labels.

thres is the maximum number of distinct tuples in the generalized relation GR.

GR is the generalized relation. Its schema is adapted from the original relation by attaching a weight to each significant attribute w_ai and adding an extra column count detailed below.

To illustrate the extraction process, we use the following relation $\mathcal{R}_{animal}$.

Table 1. $\mathcal{R}_{animal}$

Ident	Type	Height	...	place
AX34	vipera	0.3	...	France
AZ65	boa	1.2	...	Nigeria
...	...	...	...	...
AA88	ostrich	1.1	...	Australia

3.1 FAOI Algorithm

• **Procedure** FAOI(DB,Lat,FT,FL,thres):GR

```
IR = Collect_relevant_data(DB)
GR = Attribute_removal(IR, Lat)
While count(GR) > thres do
  ag = Attribute_selection(GR, Lat)
  GR = Generalization(GR, ag, FT, FL)
  GR = Merging(GR)
End
```

• **Procedure** Collect_relevant_data(DB):IR

The user enters a SQL query to collect the interesting data on which the data mining algorithm will be performed.

• **Procedure** Attribute_removal(IR,Lat):GR

```
GR = IR
For each attribute a in relation IR
do
  if FT[a] = {} or FL[a] = {} and
   Lat[a] < Distinct_values(Proj(IR, a))
  then GR = GR - <a>
end
```

This procedure examines all attributes in sequence. We remove those with no higher level concepts specified on them and whose the number of distinct values in the relation `IR` is greater than the threshold specifed in the `Lat` list. For example, the attribute *"ident"* that identifies animals in the relation $\mathcal{R}_{Animal}$ can be removed from the relation. In fact, this attribute provides no property that would characterize the class underlying the relation $\mathcal{R}_{Animal}$. This strategy is also used in DBLearn.

• **Procedure** `Attribute_selection(GR,Lat):ag`

```
max_value = 0
For each attribute a in relation GR
do
  value_nb = Distinct_values(Proj(GR, a))
  if value_nb > max_value
  and value_nb > Lat[a]
  then ag = a
    max_value = value_nb
end
```

This procedure chooses the next attribute on which the generalization operator will be applied. As in DBLearn, we select the attribute with the maximum number of distinct values.

• **Procedure** `Generalization(RG,ag,FT,FL):RG`

```
if ag is numeric attribute
then
  GR = generalization_FL(GR, ag, FL[a])
else
  GR = generalization_FT(GR, ag, FT[a])
end
```

According to the type of the selected attribute, two procedures are used for the generalization: `generalization_FL` if it is a numeric attribute and `generalization_FT` otherwise.

In this paper, we only explain `generalization_FT` but `generalization_FL` can be defined in the same way.

Before that, we explore several problems we encountered while using fuzzy thesauri to represent background knowledge.

The first difference with the DBLearn system is that, in databases, tuples can contain values which are not necessarily leafs of the thesaurus. For example, a database user observing a snake, does not always have necessarily information to identify to which type it belongs. If there exist different levels in the initial relation, we have to control the process applying the generalization one level at a time to avoid over-generalization.

The generalization is applied on concepts with maximum depth and climbs one level at each step. For example, table 2 contains different level concepts. In the first generalization step on the attribute "type", only the values *Ostrich* and *Albatroos* will be generalized because their depth level is the maximum in the relation (cf figure 1). This maximum defines the first generalization level, which decreases at each next step.

Table 2. An extract of the relation GR

type	wt	height	wh	...	count
ostrich	1	tall	0.9	...	30
albatross	1	tall	0.7	...	40
snake	1	small	0.8	...	10
...	...	...	...	...	...

Table 3. Table 2 after one generalization and merging

type	wt	height	wh	...	count
bird	1	tall	0.78	...	70
snake	1	small	0.8	...	10
...	...	...	...	...	...

The second difference with the DBLearn system is the multi-generalization and the use of different kinds of links to represent background knowledge, particularly the link "is not". A concept can be generalized into more than one concept or a concept can not be generalized into a higher level concept. Mistakes can be made, if we do not take some precautions. Especially, the generalization process should not leave out the previous steps as it is made in DBLearn. For example, if a generalization is performed on the attribute "type", table 3 describes the relation after merging identical tuples. When identical tuples are merged, the new weight associated to each attribute is calculated with a weighted average (cf `Merging` procedure).

After a second generalization stage on table 3, table 4 is obtained. Instead of table 4, we must obtain the relation described in table 5.

Table 4. Table 3 one step after

type	wt	height	wh	...	count
flying animal	0.8	tall	0.78	...	70
oviparous	1	tall	0.78	...	70
oviparous	0.9	small	0.8	...	10
...	...	...	...	...	...

Table 5. Table 3 one step after, respecting the link "is not" in the thesaurus

type	wt	height	wh	...	count
flying animal	0,8	tall	0,7	...	40
bird	1	tall	0,9	...	30
oviparous	1	tall	0,78	...	70
oviparous	0,9	small	0,8	...	10
...	...	...	...	...	...

To ensure the validity of the generalization, a particular procedure must be applied to process the multi-inheritance and the "is not" link cases.

The multi-generalization can generate problems only when several paths exist between a specific concept *cs* and a generic concept *cg*. So, if we adopt the strategy of generalizing at each step, *cs* will be generalized as many times as the number of paths relating it to *cg*. The strategy we adopt is to generalize *cs* directly to *cg*.

To solve the problem generated by the "is not" link, we adopted a similar strategy. We detect concepts related with this link. If there exists a link labelled "is not" between a concept *cs* and a generic concept *cg*, the concept *cs* can not be generalized into *cg*'s ancestors. *cs* is generalized into the direct *cg*'s descendants that are also ancestors of *cs*. But, this attribute value can not be generalized next and it can not be merged with other tuples. That is why the tuple representing an exception is stored in a particular relation named `ER`. These "exceptional" tuples can be generalized on other attributes values and they can be merged together. Then, the `ER` relation also stores information for each attribute indicating if the attribute value can be generalized in the future (cf table 7) or not.

For example, in the thesaurus (cf figure 1), the concept *Ostrich* is related to the concept *Flying Animal* with a link "is not". So when table 2 is generalized on the first column, *Ostrich* is directly generalized into *Bird* and this new tuple can not have new generalizations on this value afterwards. *Ostrich* is also generalized in *Oviparous* and this new tuple can handle the process further on.

Table 6. Table 2 after a generalization

type	wt	height	wh	...	count
bird	1	tall	0.7	...	40
oviparous	0.7	tall	0.9	...	30
snake	1	small	0.8	...	10
...	...	...	...	...	...

Table 7. ER table generated by generalizing table 2 into table 6

type	wt	gt	height	wh	...	count
bird	0.7	false	tall	1	...	30

gt indicates if the value *Bird* can be generalized next.

The generalized relation `GR` is divided into two relations: `ER` (cf table 7) and `NR` (cf table 6). The procedure of generalization and merging on each relation can be processed concurrently. The generalized table `GR` is obtained by an union of the relation `ER` and `NR` after generalization and merging.

- **Procedure:** `generalization_FT(ER,NR,ag,ft): NR, ER`

```
ER_generalization_FT(ER,ag,Lat,ft): ER
NR_generalization_FT(NR,ag,LAt,ft): NR, NER
ER = Union(ER, NER)
```

The generalization `GR` is obtained by an union of the relation `ER` and `NR`.

```
GR = Union(NR, Proj(ER,a1,wa1,...,an,wan,count))
```

The function `general_concepts`, used below, gives for a concept (and its associated weight) the set of more general concepts taking into account problems introduced by the multi-generalization and the link "is not". Using the exemple of the table 2, `general_concepts(thes1,ostrich,1,1)`, where the third parameter represents the weight associated and the fourth the level set by the user in the list `Lat`, returns the following values: `(bird, 0.7,false)` and `(oviparous, 0.7, true)`. The level set by the user indicates in the procedure if problems may occur or not in the generalization process. For example, still using the table 2, if the level set by the user is 2, `general_concepts(thes1,ostrich,1,2)` returns only `(bird,0.7,true)`. Any problem can occur because the concept related with *Ostrich* by a link "is not" is at the level 1 in the thesaurus (cf figure 1). The generalization can not climb to this value.

There are actually two generalization procedures, that are slightly similar. They only differ in their target relation. `NR_generalization_FT` addresses relation `NR`, and `ER_generalization_FT` addresses relation `ER`. We describe them below.

• **Procedure:** `ER_generalization_FT(ER,ag,Lat,ft): ER`

```
For each t in relation ER do
 if generalizable(ft,t.ag)
 then
  Lcg = general_concepts(ft,t.ag,t.w_ag,Lat[ag])
  if Lcg = {}
  then t = update(t,g_ag,false)
   ER = update_r(ER,t)
  else
   For each e in list Lcg do
    if e.g = false
    then t = update(t,g_ag,false)
    end
    t = update(t,ag,e.cg)
    t = update(t,w_ag,e.wcg)
    ER = update_r(ER,t)
```

• **Procedure:** `NR_generalization_FT(NR,ag, Lat,ft): NR,ER`

```
NNR = {}
For each t in relation NR do
 if generalizable(ft,t.ag)
 then
  Lcg = general_concepts(ft,t.ag,t.w_ag,Lat[ag])
  if Lcg = {}
  then te =<t.a1,t.w_a1,true,...>
   te = update(te,t.g_ag,e.g)
   ER = ER + <t>
  else
```

```
   For each e in list Lcg do
    if e.g = false
    then te =<t.a1,t.w_a1,true,...>
     te = update(te,t.ag,e.cg)
     te = update(te,t.w_ag,e.w_cg)
     te = update(te,t.g_ag,e.g)
     ER = ER + <t>
    else t = update(t,ag,e.cg)
     t = update(t,w_ag,e.w_cg)
     NNR = NNR + <t>
NR = NNR
```

• **Procedure:** `Merging(GR):GR`

```
ER = Merging(ER)
NR = Merging(NR)
GR = Union(NR, Proj(ER, a1,wa1,...,an,wan,count))
```

The `Merging` procedure applied on the relation gathers identical tuples together. Two tuples are considered as being identical if they have the same value for all the columns except for `w_ai` and `count`. In the merged tuple, the weight w_ai associated to an attribute value depends on the attribute weights $t_k.w\_ai, i \in \{1..n\}$ and the coverage $t_k.count$ of all the N tuples t_k to be merged. It is calculated as follows:

$$w\_ai_{(t_1,\ldots,t_N)} = \frac{\sum_{k\in\{1,..,N\}} t_k.w\_ai \times t_k.count}{\sum_{k\in\{1,..,N\}} t_k.count}$$

The factor $t_k.count$ allows to take into account the contribution of each tuple to the merged tuple, from a representativity point of view. The division by the sum $\sum_{k\in\{1,..,N\}} t_k.count$ normalizes the calculated weight. Finally, and as the generalization procedure, the merging procedure can be performed concurrently on each of the relations `ER` and `NR`.

3.2 Generating Rules

The interpretation stage consists of giving a set of rules to the user. A rule takes the following form:

Q are $concept_1$ (d_1) and $concept_2$ (d_2) and ...

where Q is a fuzzy quantifier, $concept_i$ are concepts (attribute values) and d_i are satisfaction degrees of concepts.

To be more useful and expressive to the user, we translate the rules into sentences. For example, the following sentence express such a rule: "most animals are apparently snakes and somewhat small". To achieve this interpretation, we procede in two steps:

First step: Final ER-NR merging. We apply the merging procedure on the relation `GR` (i.e. Merging(`ER` + `NR`)).

Second step: interpretation of `ER` and `NR`.

The process examines all tuples of `ER` and `NR` in sequence. For a tuple (c1, w1, ..., cn, wn, count) we can deduce the rule: "count% of relevant data are c1 (w1) and ... and cn (wn)" where count% is the percent of count with regard to the number of tuples in the initial relation (containing relevant data).

Next, a rule is converted to a sentence using a quantifier and linguistic terms to qualify each element of the rule. The quantifier Q is translated into a term taken from a given set of terms [7]. In this set, each term is represented by a fuzzy set. In the same way, the different degrees become fuzzy terms modulating the concepts. We shall not go further in this process but we illustrate it in the following example. Let table 8 be the result of the extraction process from an initial relation containing 1000 european animals.

Table 8. Final relation from an extraction process

type	wt	height	wh	...	count
oviparous	0.9	small	0.8	...	800
oviparous	1	medium	0.9	...	200
flying animal	1	small	0.9	...	200

Three rules can be deduced from this relation:
80% of european animals are oviparous (0.9) and small (0.8)
20% of european animals are oviparous (1) and medium (0.9)
20% of european animals are flying animals (1) and small (0.9)
These three rules can have the following linguistic interpretation:
"Most european animals are roughly oviparous and roughly small"
"Few european animals are oviparous and roughly medium"
"Few european animals are flying animals and roughly small"

4 Conclusion and Further Work

We presented a fuzzy attribute-oriented induction method for rules discovery in relational databases. This method adapts the one proposed in the DBLearn [2] system to handle background knowledge represented by fuzzy thesauri and fuzzy labels. Using this kind of models allowed us to take into account the inherent imprecision and uncertainty of the domain knowledge representation. Fuzzy thesauri also offers a framework for representing graduations in the generalization process. In short, valuation of links combined with "is not" relationship, and multiple inheritance enrich attribute-oriented induction methods and offer an easy way to represent exceptions.

Finally, we consider a concrete extension of this work by investigating the *data summary* issue through the construction of a fuzzy data synthesis system based on a concept formation algorithm [10].

References

1. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. Uthurusamy, R. (eds): Advances in Knowledge Discovery and Data Mining. *AAAI Press/ the MIT Press*, ch. 1, pp. 1–37 (1996) 1
2. Han, J., Cai, Y., Cercone, N.: Knowledge discovery in databases: An attribute-oriented approach. *18th Very Large DataBases Conference*, (Vancouver, Canada), pp. 547–559 (1992) 1, 12
3. Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Xia, A.: Dbminer : a system for mining knowledge in large relational databases. *2nd International conference on knowledge discovery in databases and data mining*, (Portland, Usa), pp. 250–255 (1997) 1
4. Subtil, P., Mouaddib, N., Foucaut, O.: Fuzzy thesaurus for databases. *6th International Fuzzy Systems Association World Congress - Vol. 2*, (Sao Paulo, Brazil), pp. 349–352, (1995) 2
5. Zadeh, L.: Concept of a linguistic variable and its application to approximate reasoning (1). *Information Systems*, Vol. 8, pp. 199–249 (1975) 4
6. Bosc, P., Dubois, D., Prade, H.: Approximate data reduction and fuzzy functionnal dependencies. *6th International Fuzzy Systems Association World Congress - Vol. 2*, (Sao Paulo, Brazil), pp. 369 – 371, (1995) 4
7. Yager, R.: Linguistic summaries as a tool for database discovery. *FUZZ-IEEE'95 Worskshop on Fuzzy Databases Systems*, (Yokohama, Japan), pp. 79–84, (1995) 12
8. Michalsky, R.S., Stepp, R.: Learning from observation : conceptual clustering. Michalsky, R.S., Carbonell, J.G. and Mitchell T.M. eds, *Machine Learning : an Artificial Intelligence Approach*, Vol. 1, ch. 11, pp. 331–363 (1983) 4
9. Quinlan, J.R.: Induction of decision trees. *Machine Learning*, pp. 81–106 (1986) 5
10. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. *Machine Learning*,Vol. 2, pp. 139–172 (1987) 5, 12
11. Gennari, J.H., Langley, P., Fisher, D.H.: Models of incremental concept formation. In *Knowledge Acquisition and Learning*, Buchanan, B.G. and Wilkins, D.C. (eds) (1989) 5
12. Kodratoff, Y., Diday, E.: Induction symbolique et numérique à partir des données. Cépaduès-Editions (1991) 4

Fast Methods with Magic Sampling for Knowledge Discovery in Deductive Databases with Large Deduction Results

Chien-Le Goh, Masahiko Tsukamoto, and Shojiro Nishio

Department of Information Systems Engineering
Graduate School of Engineering, Osaka University
2-1 Yamadaoka, Suita, Osaka 565, Japan
Tel: +81-6-879-7820, Fax: +81-6-877-9463
{tuka,nishio}@ise.eng.osaka-u.ac.jp

Abstract. The ability of the *deductive database* to handle *recursive queries* is one of its most useful features. It opens up new possibilities for users to view and analyze data. This ability to handle recursive queries, however, still cannot be fully utilized because when *recursive rules* are involved, the amount of deduced facts can become very large, making it difficult and sometimes impossible to store, view or analyze the query results. In order to overcome this problem, we have proposed the DSK method and the DSK(S) method to discover *characteristic rules* from large amount of deduction results without having to store all of them. In this paper, we propose two new methods, the DSK(T) method and the DSK(ST) method which are faster than the DSK method and the DSK(S) method respectively. In addition, we propose a new sampling method called *magic sampling*, which is used by the two methods to achieve the improvement in speed. Magic sampling works when *linear recursive rules* are involved and the magic set algorithm is used for deduction.

1 Introduction

The ability of the deductive database to handle recursive queries is one of its most useful features. It opens up new possibilities for users to view and analyze data. For example, a user who has stored data about all the vertices and arcs of a graph in a deductive database can easily instruct the database management system to deduce all the possible paths from a certain vertex and analyze them. This ability to handle recursive queries, however, still cannot be fully utilized because when recursive rules are involved, the amount of deduced facts can become very large, making it difficult and sometimes impossible to store, view or analyze the query results.

In [10], we have proposed five methods of applying the attribute-oriented algorithm proposed by J. Han, Y. Cai, and N. Cercone[11,12] to discover characteristic rules in large amount of deduced facts, deduced by recursive or non-recursive rules, without having to store all of them. We chose the algorithm

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 14–29, 1999.

because it has been found to be effective as a data mining [8,9] algorithm in its implementation in DBLEARN [13]. In addition, the use of the algorithm has been extended to object-oriented databases in [14]. In [10], we view the extensional database (EDB) of a deductive database as a relational database, and apply the algorithm to deductive databases.

Comparisons among the five methods (KD, SDK, DK, DSK, DSK(S)) in terms of speed, accuracy, memory requirement, etc. have been made. From the comparisons, we realized that the two most useful methods are the DSK method and the DSK(S) method. The DSK method is reasonably fast and accurate while the DSK(S) method is slightly slower, uses a lot less memory and accurate. "D", "S", and "K" stand for "deduction", "sampling", and "knowledge discovery" respectively. The "S" in brackets stands for "space".

In this paper, we propose two new methods, the DSK(T) method and the DSK(ST) method ("T" stands for time) which are faster than the DSK method and the DSK(S) method respectively. In addition, we propose a new sampling method called *magic sampling*, which is used by the two methods to achieve the improvement in speed. Magic sampling works when linear recursive rules [4] are involved and the magic set algorithm [3,4,6] is used for deduction.

The organization of this paper is as follows. In section 2, an example which will be used in the explanation is presented. The DSK(T) method and the DSK(ST) method are explained in sections 3 and 4. Qualitative comparisons among the DSK method, the DSK(S) method, the DSK(T) method, and the DSK(ST) method are presented in section 5. Section 6 is the conclusion of this paper.

2 An Example

For the purpose of explanation, let us suppose an imaginary case where we have the medical records for the last one thousand years of all the people in Japan who suffered or are suffering from at least a kind of allergies. A one thousand year period is supposed because we need medical records that cover many generations to act as a good example. The medical records are stored in deductive databases distributed all over Japan but they can be accessed as a single large database from a user's point of view. Here, a deductive database is regarded as a relational database with deduction mechanism.

The extensional database of the imaginary deductive database has the following scheme:

```
MEDICAL_RECORD(Patient, Parent, Allergy)
```

Although a more realistic medical record has more attributes such as address, height, weight, etc., we only assume three attributes in this example to make things simple. Based on the above scheme, we have a relation such as the one in Fig. 1 to store the extensional database of the deductive database.

In the intensional database (IDB), we assume that rules to deduce the ancestors of patients exist and are expressed as below. In the rules, $parent(R, S)$

Patient	Parent	Allergy
Masako YAMADA	Taro YAMADA	Y
Masako YAMADA	Tomoko YAMADA	Y
Taro YAMADA	Toru YAMADA	X
Taro YAMADA	Keiko YAMADA	X
Shigeo YAMADA	Masako YAMADA	Z
Hiroaki SUZUKI	Jiro SUZUKI	Z
⋮	⋮	⋮

Fig. 1. `MEDICAL_RECORD(Patient, Parent, Allergy)`.

Patient	Ancestor
Taro YAMADA	Toru YAMADA
Taro YAMADA	Keiko YAMADA
Masako YAMADA	Toru YAMADA
Masako YAMADA	Keiko YAMADA
Masako YAMADA	Taro YAMADA
Masako YAMADA	Tomoko YAMADA
Hiroaki SUZUKI	Jiro SUZUKI
⋮	⋮

Fig. 2. A deduced relation with `Patient` and `Ancestor`.

is equivalent to `MEDICAL_RECORD(`$R, S,$ `-)`. Here, "`-`" means "not specified" and anything can be inserted into "`-`".

$$ancestor(R, S) :- parent(R, S).$$
$$ancestor(R, S) :- ancestor(R, T), parent(T, S).$$

Using the rules for deducing ancestors, it is possible to deduce all the ancestors of any number of patients (Fig. 2). From the deductive database, we wish to find a previously unknown hereditary allergy. The search can be conducted by finding out the characteristic rules of all the allergies throughout the generations.

In order to use the attribute-oriented algorithm to find the characteristic rules, we need some background knowledge in the form of concept hierarchies. In our example, we assume that we have a concept hierarchy like the one shown in Fig. 3. The two layers from the bottom of the concept hierarchy can be easily constructed from the extensional database using the patients' names and the allergies they suffered from.

The concept hierarchy is used to generalize the names of `Patient` and `Ancestor` to the allergies they suffered from and then to broader terms that describe the allergies. Not all the names can be generalized because we do not know the allergies that some of the patients' parents suffered from. However,

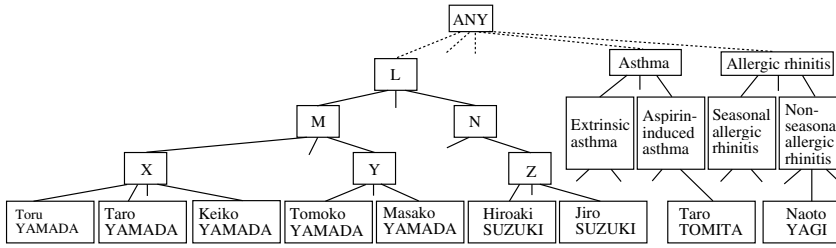


Fig. 3. A concept hierarchy used as background knowledge.

Patient	Ancestor	Count	Percentage
M	M	74	37.0%
N	N	18	9.0%
N	O	19	9.5%
⋮	⋮	⋮	⋮

Fig. 4. Characteristic rules of the deduced relation.

most can be generalized and this should be sufficient for our purpose of finding characteristic rules.

If we randomly sample a certain number of tuples, for example two hundred tuples, from the deduced relation which contains attributes `Patient` and `Ancestor` and generalize the instances of `Patient` and `Ancestor` using the concept hierarchy in Fig. 3, we may obtain the characteristic rules of the deduced relation in the form shown in Fig. 4 depending on the attribute thresholds and the table threshold values set by the user. The characteristic rules can then be processed using probabilistic and statistical methods to find hereditary allergies. The errors between characteristic rules obtained from random samples and the whole amount of deduced facts are very small. This has been shown in [15].

The way that we used above can help us find the characteristic rules. However, since the deduced facts of every patient's ancestors are often too many to be stored in the limited memory capacity we have, we are not able to obtain a random sample. We need to find a way to obtain a random sample and learn the characteristic rules without having to store all the deduced facts. The DSK(T) method and the DSK(ST) method represent two solutions.

3 The DSK(T) Method

If we take a closer look at the DSK method, we can find some redundancies in processing when we try to build the *aggregate relation* introduced and used by the DSK method. For example, when we find the number of ancestors of Masako

YAMADA, we can also find the number of ancestors of Taro YAMADA; but that is not done in the DSK method. The DSK method finds the number of ancestors of Masako YAMADA and Taro YAMADA independently from scratch.

The DSK(T) method aims to increase the speed of forming the aggregate relation by utilizing the information of each query. However, it works only when linear recursive rules are involved and when the magic set algorithm is used for query processing. These restrictions are not too harsh for real life problems because there is a belief that most "real life" recursive rules are linear [4] and the magic set algorithm is popular and efficient. With the restrictions, we obtain the following lemma. We call a sampling method using the lemma magic sampling.

Lemma 1 $\forall x, x \in magic(a) \Rightarrow magic(x) \subseteq magic(a)$;
a is a constant and $magic(a)$ *is the magic set of* a.

Proof sketch. A linear recursive rule has the following form.

$$P_0 : -P_1, P_2, P_3, ..., P_m, P_0. \quad \cdots\cdots (1)$$

For simplicity, variables have been omitted. Each literal may have different *arities* [5] but P_0 at the head and P_0 in the body must have the same arity.

From rule (1), adorned rules where only one variable is bound have the form shown below. (The lemma stands when only one variable is bound.)

$$P_0^{..ffbff..} : -P_1, P_2, P_3, ..., P_m, P_0^{..ffbff..}. \quad \cdots\cdots (2)$$

Although there are many ways to generate magic rules from adorned rules in the form of (2), they can be summarized into the following two forms.

$$magic.P_0^{..ffbff..}(X) : -P_i, P_{i+1}, ...P_j, magic.P_0^{..ffbff..}(X).$$

$$magic.P_0^{..ffbff..}(Y) : -P_k, P_{k+1}, ...P_l, magic.P_0^{..ffbff..}(X).$$

The first form shows that for a constant a,

$$magic.P_0^{..ffbff..}(a) = \{a\} \text{ or } \emptyset$$

and the lemma is correct. The second form is a rule of transitive closure and in this case too, the lemma is correct. □

This lemma means when using the magic set algorithm to process a query q involving a constant a, the magic set of a can also be used to process a query q involving x. For example, we can use $magic$(Masako YAMADA) to find the ancestors of Masako YAMADA, Taro YAMADA, etc. Finding the magic set is "expensive" and by eliminating the need to find $magic(x)$ for each x, we can have a significant gain in speed.

The DSK(T) method consists of three steps: (i) deduction, (ii) sampling and (iii) knowledge discovery. At step 1, deduction is performed and the number of deduced facts for each fact in the extensional database is calculated. Based

Patient	Number of ancestors
Masako YAMADA	1023

Fig. 5. The aggregate relation. (Step 1a)

Patient	Number of ancestors
Masako YAMADA	1023
Taro YAMADA	255
Toru YAMADA	200
Keiko YAMADA	55
⋮	⋮

Fig. 6. The aggregate relation. (Step 1b)

on the example in section 2, we look at the extensional database and select the first tuple Masako YAMADA. Then we find the magic set of Masako YAMADA, *magic*(Masako YAMADA), and find her total number of ancestors (see Fig. 5).

After that, for each person x in *magic*(Masako YAMADA), we use *magic*(Masako YAMADA) to find the number of ancestors of x. Each x is checked against the aggregate relation to see whether he or she has appeared in it or not before we find the number of ancestors of x. If he or she has already appeared, we skip and go to the next x. After finding the number of ancestors of each x, we may obtain something like Fig. 6.

Then we go to the second patient, Taro YAMADA, in the extensional database. By checking the aggregate relation, we know that his total number of ancestors has been calculated. So we skip and go to the third patient, Shigeo YAMADA.

Since Shigeo YAMADA is not in the aggregate relation, we have to find the magic set of Shigeo YAMADA and calculate his total number of ancestors. Then again for each person x in *magic*(Shigeo YAMADA), we check whether he or she has already appeared in the aggregate relation before we find the total number of ancestors of x. We find that every x has already appeared and thus can skip the calculation.

The above process continues until we have calculated the total number of ancestors of every patient in the extensional database. Finally, the accumulated total number of ancestors of all the patients is calculated and we should obtain an aggregate relation like Fig. 7.

At step 2, sampling is performed in 2 stages. At the first stage, a patient is selected according to the probability p where

$$p = \frac{the\ number\ of\ ancestors\ of\ a\ patient}{the\ number\ of\ ancestors\ of\ all\ the\ patients}.$$

Patient	Number of ancestors
Masako YAMADA	1023
Taro YAMADA	255
Toru YAMADA	200
Keiko YAMADA	55
Shigeo YAMADA	1024
⋮	⋮
Hiroaki SUZUKI	2000
⋮	⋮

Fig. 7. The aggregate relation. (End of step 1)

Then at the second stage, all the selected patient's ancestors will be deduced and one of them, which is equivalent to one tuple from the deduction results, will be sampled by random. After a tuple is sampled, the number of ancestors of the selected patient and the total number of ancestors of all the patients are each deducted by one. The two stages of step 2 are repeated many times until a user specified n number of tuples have been sampled.

The sample is passed on to step 3 where the process of discovering characteristic rules is performed using the attribute-oriented algorithm and we may obtain a relation like Fig. 4. Steps 2 and 3 of the DSK(T) method are the same as steps 2 and 3 of the DSK method. Further details of the DSK method can be found in [10].

Summarizing the DSK(T) method, we can obtain the following algorithm:

Algorithm 1. *Discovery of characteristic rules using the DSK(T) method.*

Input. (i) A set of concept hierarchies, (ii) a set of attribute thresholds, (iii) a table threshold, (iv) an intensional database, (v) an extensional database, and (vi) n, the number of tuples to obtain to form a sample.
Output. A set of characteristic rules of the deduced facts.

Method.

1. Collect the set of task-relevant data from the extensional database by a database query.
2. Create a view v of the attributes which bind the arguments of IDB predicates used in deduction from the set of task-relevant data.
3. **begin**
   for each tuple s **in** v **do** {
   if s is not in the aggregate relation R
   then do {
   find $magic(s)$,
   use $magic(s)$ to calculate the total number of tuples N_t that can be deduced from s,
   store s and N_t in R,
   delete the deduced tuples of s,
   for each $x \in magic(s)$ **do** {
   if x is not in R
   then do {
   use $magic(s)$ to calculate the N_t of x,
   store x and N_t in R,
   delete the deduced tuples of x }
   }
   delete $magic(s)$ }
   }
   Calculate the sum S of all N_t.
   end

4. **do** {
   Sample a tuple t from relation R based on the formula $p(t) = \frac{N_t}{S}$,
   deduce from t,
   select a result u by random from all the deduction result that can be deduced from t,
   store t (without N_t) and u together as one tuple in relation Q,
   reduce N_t by one,
   reduce S by one,
   delete all deduction results }
   until n distinct tuples have been stored in Q.
   - Comment:
   $p(t)$ is the probability for t to be selected from R.
   Distinct tuples can be obtained by avoiding deduction result selected previously.
   If this does not work, select a new t.
5. Generalize relation Q according to the set of attribute thresholds, the table threshold and the set of concept hierarchies provided.
6. Present the generalized relation as a set of characteristic rules with two additional attributes, `Count` and `Percentage`.

Like the DSK method, we have the following theorem:

Theorem 1 *The distribution of a sample obtained by the DSK(T) method from deduction results D is the same as the distribution of a sample obtained by random sampling from D.*

Proof sketch. Assuming that the total number of deduction results is S, the probability that a deduction result is sampled is $\frac{1}{S}$. Using the DSK(T) method, the probability that a deduction result is sampled is $\frac{N_t}{S} \times \frac{1}{N_t}$, which is equal to $\frac{1}{S}$. □

4 The DSK(ST) Method

The DSK(S) method also has some redundancies in processing when building the *buffer relation*. In this section, we propose the DSK(ST) method which does away with the redundancies and improves the speed of the DSK(S) method.

The DSK(ST) uses the idea of magic sampling to improve the DSK(S) method and has the same restrictions as that of the DSK(T) method, i.e. the DSK(ST) method can be applied when only linear recursive rules are involved and the magic set algorithm is used for query processing. With the restrictions, **Lemma 1** also stands true for the DSK(ST) method.

The DSK(ST) method consists of 2 steps: (i) deduction and sampling, and(ii) knowledge discovery. At the first step, deduction and sampling are performed together to obtain a random sample of n tuples from the deduction results. n is specified by the user. Step 1 consists of three stages. At stage 1, a sample of patients is taken. At stage 2, ancestors of each patient in the sample are deduced and one of them is selected for each patient. At stage 3, more patients and their ancestors are selected if a sample of n tuples has not yet been obtained at stages 1 and 2.

Using the example in section 2, stage 1 starts by creating a view (see Fig. 8) with only the attribute `Patient` which binds the argument of the IDB predicates. Then the first patient of the view, Masako YAMADA, is selected and her total number of ancestors, t, is calculated using the magic set algorithm and stored in a variable called *scanned_tuples*. Assuming that Masako YAMADA has 1023 ancestors (see Fig. 6), it will be "$scanned\_tuples = t = 1023$".

Patient
Masako YAMADA
Taro YAMADA
Shigeo YAMADA
Hiroaki SUZUKI
⋮

Fig. 8. A view with only the attribute `Patient`.

Variable	Patient
$selected\_tuple_1$	Masako YAMADA
$selected\_tuple_2$	Masako YAMADA
$selected\_tuple_3$	Masako YAMADA
⋮	⋮
$selected\_tuple_i$	Masako YAMADA
⋮	⋮
$selected\_tuple_k$	Masako YAMADA

Fig. 9. Selected patients after examining the first patient, Masako YAMADA.

Then Masako YAMADA is stored in a relation called *buffer relation* like what is shown in Fig. 9. The buffer relation will eventually hold a sample at the end of step 1. k, which is the number of tuples in the buffer relation, is defined as an *integer* which is equal to $n \times r$ where r, the redundancy factor, is a real number specified by the user. k is purposely made to be r times greater than n so that we can still obtain a sample of at least n number of tuples from the buffer relation after avoiding similar tuples. r must be greater than 1 and the user can set it to, for example, 1.2 to obtain enough tuples to form a sample.

After that, unlike the DSK(S) method which discards the information of Masako YAMADA's ancestors, we make use of $magic$(Masako YAMADA) to find the total number of ancestors of each x where $x \in magic$(Masako YAMADA) = {Taro YAMADA, Toru YAMADA, Keiko YAMADA, ...}

Taro YAMADA in $magic$(Masako YAMADA) is first examined and the total number of ancestors, t, of Taro YAMADA is calculated. Then, for each variable from $selected\_tuple_1$ to $selected\_tuple_k$, modification of the content is determined by the following probability.

$$selected\_tuple_i = \begin{cases} selected\_tuple_i & ; \text{with probability } \frac{scanned\_tuples}{scanned\_tuples+t} \\ \text{Taro YAMADA} & ; \text{with probability } \frac{t}{scanned\_tuples+t} \end{cases}$$

Variable	Patient
$selected\_tuple_1$	Taro YAMADA
$selected\_tuple_2$	Masako YAMADA
$selected\_tuple_3$	Masako YAMADA
⋮	⋮
$selected\_tuple_i$	Taro YAMADA
⋮	⋮
$selected\_tuple_k$	Masako YAMADA

Fig. 10. Selected patients after examining Taro YAMADA.

Variable	Patient
$selected\_tuple_1$	Taro YAMADA
$selected\_tuple_2$	Masako YAMADA
$selected\_tuple_3$	Toru YAMADA
⋮	⋮
$selected\_tuple_i$	Keiko YAMADA
⋮	⋮
$selected\_tuple_k$	Masako YAMADA

Fig. 11. Selected patients after examining all the elements of *magic*(Masako YAMADA).

The variable *scanned_tuples* is also updated by incrementing it by t. Assuming that t for Taro YAMADA is 255, *scanned_tuples* will be incremented to 1278 ($= 1023 + 255$) and we may obtain something like Fig. 10.

For the rest of the patients in *magic*(Masako YAMADA), variables $selected\_tuple_1$ to $selected\_tuple_k$ and *scanned_tuples* are updated accordingly until every patient in *magic*(Masako YAMADA) has been examined (see Fig. 11). We then store the elements of *magic*(Masako YAMADA) in a relation called *cache relation* (see Fig. 12). The cache relation tells us who have already been examined and should not be examined again.

After we have finished examining the first patient and the elements of its magic set, we go to the second patient, Taro YAMADA. We first have to check to see whether Taro YAMADA is in the cache relation. Since he is in there, we do not have to find his ancestors nor update the buffer relation. We just skip Taro YAMADA and delete Taro YAMADA from the cache relation because he will not appear later when we go further down the view and we thus do not need to check for his existence again in the cache relation.

The third patient, Shigeo YAMADA, is not in the cache relation. Therefore, we have to find his magic set and his ancestors and update the buffer relation.

Patient
Taro YAMADA
Shigeo YAMADA
Hiroaki SUZUKI
⋮

Fig. 12. A cache relation

Variable	Patient
$selected\_tuple_1$	Taro YAMADA
$selected\_tuple_2$	Masako YAMADA
$selected\_tuple_3$	Hiroaki SUZUKI
⋮	⋮
$selected\_tuple_k$	Masako YAMADA

Fig. 13. Selected patients at the end of stage 1.

We then proceed to examine all the elements in the magic set of Shigeo YAMADA. For elements which are already in the cache relation, we neither find their ancestors nor update the buffer relation. They are simply deleted from the cache relation. For elements which are not in the cache relation, we use *magic*(Shigeo YAMADA) to find their ancestors, update the buffer relation and store the elements in the cache relation for future references.

The above process is repeated until we reach the last patient of the view. Let us suppose that we have obtained something like Fig. 13 after stage 1 and we have found that there are 100,000 `Patient-Ancestor` relations ($scanned\_tuples = 100000$).

At stage 2, for each selected patient, all his or her ancestors are deduced and one of them is randomly selected. The probability that an ancestor is selected is $\frac{1}{t}$ where t is the total number of ancestors. This will give us Fig. 14. From Fig. 14, we only select n number of distinct tuples to form a sample. This can be done by selecting the tuples one by one from top to bottom while ignoring the redundant tuples. For example, from Fig. 14, we may get a sample like Fig. 15, after ignoring redundant `Patient-Ancestor` relations such as the "Masako YAMADA - Toru YAMADA" relation.

If at the end of stage 2, we can find a sample of n number of tuples, then we can pass this sample to step 2 where we try to discovery of characteristic rules is carried out. However, there is a possibility that there are so many overlapping `Patient-Ancestor` relations that we cannot find enough tuples to form a sample. To solve this, we will have to go to stage 3 to find additional tuples.

Variable	Patient	Ancestor
$selected\_tuple_1$	Taro YAMADA	Keiko YAMADA
$selected\_tuple_2$	Masako YAMADA	Toru YAMADA
$selected\_tuple_3$	Hiroaki Suzuki	Jiro Suzuki
⋮	⋮	⋮
$selected\_tuple_k$	Masako YAMADA	Toru YAMADA

Fig. 14. Selected patients and their selected ancestors.

Patient	Ancestor
Taro YAMADA	Keiko YAMADA
Masako YAMADA	Toru YAMADA
Hiroaki Suzuki	Jiro Suzuki
⋮	⋮
Tadashi IZUMI	Jun IZUMI

Fig. 15. A sample obtained after stage 2.

Suppose we are short of m tuples to form a sample. At stage 3, m patients are selected using the same method as in stage 1 using variables $selected\_tuple_1$ to $selected\_tuple_l$ where l is equal to $m \times r$.

In order to be statistically correct, the total number of ancestors of each patient, t, is reduced by the total number of `Patient-Ancestor` relations that are already in the sample and involve the patient being examined. This means, for example, if there are 10 `Patient-Ancestor` relations in the sample with Taro YAMADA as the attribute value of `Patient`, then when Taro YAMADA is examined, t of Taro YAMADA will be reduced by 10 ($t = t - 10$).

After l patients have been selected, the ancestors of each selected patient is deduced and one of them is randomly selected using the same method of stage 2. Again, to be statistically correct, `Patient-Ancestor` relations which are already in the sample must be avoided. For example, if Taro YAMADA's ancestors are {Toru YAMADA, Keiko YAMADA, ...} and the relation "Taro YAMADA - Keiko YAMADA" is already in the sample, Keiko YAMADA is deleted from the set of ancestors of Taro YAMADA before the selection of ancestor is made.

At the end of stage 3, m number of distinct tuples which are not in the sample are selected from l number of `Patient-Ancestor` relations and inserted into the sample. There is a possibility that a sample with n number of `Patient-Ancestor` relations still cannot be obtained after stage 3. In this case, we will have to execute stage 3 again until we obtain n number of `Patient-Ancestor` relations.

Finally, the sample obtained at step 1 is passed on to step 2. At step 2 characteristic rules are discovered using the attribute-oriented algorithm. Depending on the attribute threshold and the table threshold values set by the user, the

characteristic rules in relevance to `Patient` and `Ancestor` can be something like Fig. 4.

Summarizing the DSK(ST) method, we can obtain the following algorithm:

Algorithm 2. *Discovery of characteristic rules using the DSK(ST) method.*

Input. (i) A set of concept hierarchies, (ii) a set of attribute thresholds, (iii) a table threshold, (iv) an intensional database, (v) an extensional database, (vi) r, the redundancy factor (positive real number), and (vi) n, the number of tuples to obtain to form a sample.
Output. A set of characteristic rules of the deduced facts.

Method.

1. Collect the set of task-relevant data from the extensional database by a database query.
2. Create a view v of the attributes which bind the arguments of IDB predicates used in deduction from the set of task-relevant data.
3. Initialize variable $scanned\_tuples$ to zero.
4. **begin**
   for each tuple s **in the view** v **do** {
   if s is in the cache relation C
   then delete s from C,
   else do {
   find $magic(s)$,
   Examine($s, magic(s)$),
   $magic\_temp = magic(s)$,
   for each element u **in** $magic(s)$ **do** {
   if u is in C
   then do {
   delete u from C,
   delete u from $magic\_temp$ }
   else Examine($u, magic(u)$) }
   $C = C \cup magic\_temp$ }
   }
   end.
   Examine($z, magic$) {
   use $magic$ to calculate t, the total number of tuples that can be deduced from z,
   if $z = x$ of the "$x - y$" pairs already in the sample
   then $t = t$ - the number of "$x - y$" pairs where $z = x$,
   delete the deduced tuples of z,
   update the contents of each variable $selected\_tuple_i$ where $i = 1, 2, \ldots, n \times r$ using the following formula

$$selected\_tuple_i = selected\_tuple_i \text{ ; with probability } \frac{scanned\_tuples}{scanned\_tuples+t} \text{ or}$$
$$selected\_tuple_i = z \text{ ; with probability } \frac{t}{scanned\_tuples+t}$$

   $scanned\_tuples = scanned\_tuples + t$ }
5. **begin**
   for the content x **of each** $selected\_tuple_i$ **(**$i = 1, 2, \ldots, n \times r$**) do** {
   Perform deduction,
   if deduction result $d = y$ of the "$x - y$" pairs already in the sample
   then remove d from D, the set of deduction results,
   randomly select a deduction result y from D,
   form an "$x - y$" pair and replace $selected\_tuple_i$ with the "$x - y$" pair }
   end.
6. Select n number of distinct (ignore redundant pairs) "$x - y$" pairs from $selected\_tuple_i$ and append them to relation Q.
7. **if** n number of distinct "$x - y$" pairs cannot be obtained,
   then do step 3 and step 6 to obtain additional pairs, setting n to the number of pairs needed
   until n number of distinct "$x - y$" pairs have been obtained.
8. Generalize relation Q according to the set of attribute thresholds, the table threshold and the set of concept hierarchies provided.
9. Present the generalized relation as a set of characteristic rules with two additional attributes, `Count` and `Percentage`.

The following theorem shows that the distribution of the sample obtained by the DSK(ST) method is the same as the distribution of the sample obtained by the method used in the example in section 2.

Theorem 2 *The distribution of a sample obtained by the DSK(ST) method from deduction results D is the same as the distribution of a sample obtained by random sampling from D.*

Proof sketch. Assuming that the total number of tuples of D is S, the probability that a tuple is sampled is $\frac{1}{S}$. Using the DSK(ST) method, the probability p_1 that the ath tuple in the buffer relation will be selected at the end of step 4 in Algorithm 2 is shown below. In the equation, t_a is the number of deduction results of the ath tuple and $scanned\_tuples$ is the number of tuples that have been scanned until the $(a-1)th$ tuple.

$$p_1 = \frac{t_a}{scanned\_tuples + t_a} \times \frac{scanned\_tuples + t_a}{scanned\_tuples + \sum_{i=a}^{a+1} t_i} \times \cdots$$
$$\times \frac{scanned\_tuples + \sum_{i=a}^{k-2} t_i}{scanned\_tuples + \sum_{i=a}^{k-1} t_i} \times \frac{scanned\_tuples + \sum_{i=a}^{k-1} t_i}{scanned\_tuples + \sum_{i=a}^{k} t_i}$$
$$= \frac{t_a}{scanned\_tuples + \sum_{i=a}^{k} t_i} = \frac{t_a}{S}$$

At step 5, a deduction results of the ath tuple will be selected with the probability $\frac{1}{t_a}$. Therefore, we can prove that a deduced fact is selected with the probability p_2 where

$$p_2 = \frac{t_a}{S} \times \frac{1}{t_a} = \frac{1}{S}.$$ □

5 Comparisons

We can compare the DSK method, the DSK(T) method, the DSK(S) method and the DSK(ST) method in terms of the accuracy of the characteristic rules discovered, the speed of discovery, the amount of storage needed in the process of discovery, and the ease of implementation. Because of page limitation, we only mention briefly the comparison results.

The distribution of the sample obtained by any of the four methods is the same as the distribution of the sample obtained by random sampling. Therefore, they have the same accuracy.

In terms of speed, the DSK(T) method is the fastest. The DSK(ST) method has to do more complex processing than the DSK(T) method and is therefore the second fastest. The third fastest method is the DSK method. The slowest method is the DSK(S) method. However, it should be noted that the speed improvement of the DSK(T) method and the DSK(ST) method is accomplished by narrowing the problem domain of the DSK method and the DSK(S) method.

The DSK(S) method uses the least amount of memory. It is followed by the DSK(ST) method, the DSK method and the DSK(T) method. An interesting relation between memory usage and accuracy exists in the case where the amount of available memory is fixed. The method which uses less memory can collect a larger sample and this can increase the accuracy of the characteristic rules

In terms of ease of implementation, the DSK method is the easiest to implement. The hardest to implement method is the DSK(ST) method. The DSK(S) method and the DSK(T) method lie in between and they are equally complex and equally hard to implement.

The characteristics of the four methods are summarized in Fig. 16. From the table, we can see that each method has its own unique advantage. The DSK method is best when fast implementation is needed. The DSK(S) method and the DSK(T) method is good for saving space and time respectively. The DSK(ST) method saves both space and time.

Method	DSK	DSK(S)	DSK(T)	DSK(ST)
Accuracy	A	A	A	A
Speed of discovery*	B	B	A	A-
Memory usage‡	B	A	B	A-
Implementation	A	B	B	C

∗ : When the IDB predicates involved have 3 or more arguments, the speed decreases.
‡ : When the IDB predicates involved have 3 or more arguments, memory usage increases.

Fig. 16. Qualitative comparisons among the four methods.

6 Conclusion

In this paper, we have proposed the DSK(T) method and the DSK(ST) method that improve the DSK method and the DSK(S) method in terms of speed. The two methods use our proposed sampling method, magic sampling, to achieve the improvement in speed.

Qualitative comparisons among four methods, the DSK method, the DSK(S) method , the DSK(T) method, and the DSK(ST) method have been made and the role of each method has been pointed out. We feel that quantitative comparisons based on actual experiments are still needed to verify and to provide rigorous comparisons. Furthermore, investigation to see whether these methods can be further improved is still needed.

Other than characteristic rules, we think that discovering association rules [1,2] and classification rules [7,11] from large amount of deduction results are also useful and should be further investigated. As the generation of random sample in the DSK(T) method and the DSK(ST) method are not tightly coupled with the attribute-oriented algorithm used, modifications can be easily made to accomodate other data mining algorithms. Furthermore, possibilities of further improvements when using other query processing strategies should also be investigated.

We have used a typical example of analyzing the data of patients and their ancestors to find hereditary allergies in this paper. Apart from the medical field, the methods proposed are also useful in facilitating advanced analysis of data in other fields such as in anthropology and sociology to analyze the diet of people of the *same generation* [4] and their ancestors, and in fault analysis systems and circuit simulation systems where expert systems are involved.

References

1. R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. of the ACM SIGMOD International Conference on Management of Data*, vol. 22, no. 2, pp. 207-216, 1993. 28
2. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. of the 20th VLDB Conference*, pp. 487-499, 1994. 28

3. F. Bancilhon, D. Maier, Y. Sagiv, and J. D. Ullman, "Magic Sets and Other Strange Ways to Implement Logic Programs," *Proc. of the fifth ACM SIGMOD-SIGACT Symposium on Principles of Database Systems*, pp. 1-15, 1986. 15
4. F. Bancilhon and R. Ramakrishnan, "An Amateur's Introduction to Recursive Query Processing Strategies," *Proc. of ACM SIGMOD'86*, vol. 15, no. 2, pp. 16-52, 1986. 15, 18, 28
5. J. Barwise and J. Etchemendy, *The Language of First-order Logic*, 3rd Edition, Revised and Expanded, CSLI, 1992. 18
6. C. Beeri and R. Ramakrishnan, "On the Power of Magic," *Proc. of the sixth ACM SIGMOD-SIGACT Symposium on Principles of Database Systems*, pp. 269-283, 1987. 15
7. Y. Cai, N. Cercone, and J. Han, "Attribute-Oriented Induction in Relational Databases," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley (Eds.), AAAI Press/The MIT Press, pp. 213-228, 1991. 28
8. U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery: An Overview," in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Eds.), AAAI Press / The MIT Press, pp. 1-34, 1996. 15
9. W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, "Knowledge Discovery in Databases: An Overview," in *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley (Eds.), AAAI Press / The MIT Press, pp. 1-27, 1991. 15
10. C. Goh, M. Tsukamoto, and S. Nishio, "Knowledge Discovery in Deductive Databases with Large Deduction Results: The First Step," *IEEE Trans. on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 952-956, 1996. 14, 15, 20
11. J. Han, Y. Cai, and N. Cercone, "Knowledge Discovery in Databases: an Attribute-Oriented Approach," *Proc. of the 18th VLDB Conference*, pp. 547-559, 1992. 14, 28
12. J. Han, Y. Cai, and N. Cercone, "Data-Driven Discovery of Quantitative Rules in Relational Databases," *IEEE Trans. on Knowledge and Data Engineering*, vol. 5, no. 1, pp. 29-40, 1993. 14
13. J. Han, Y. Fu, Y. Huang, Y. Cai, and N. Cercone, "DBLearn: A System Prototype for Knowledge Discovery in Relational Databases," *Proc. of ACM SIGMOD'94*, vol. 23, no. 2, pp. 516, 1994. 15
14. S. Nishio, H. Kawano, J. Han, "Knowledge Discovery in Object-Oriented Databases: The First Step," *Proc. of the AAAI Knowledge Discovery in Databases Workshop 1993*, pp. 299, 1993. 15
15. H. Kawano, K. Sonoo, S. Nishio, and T. Hasegawa, "Accuracy Evaluation of Rules Derived from Sample Data in VLKD," *Proc. of the ORSA/TIMS Joint National Meeting*, p.144, Anaheim, California, U.S.A., Nov.3-6, 1991. 17

A Heuristic Method for Correlating Attribute Group Pairs in Data Mining

Chua Eng Huang Cecil[1], Roger H.L. Chiang[1], and Ee-Peng Lim[2]

[1] Information Management Research Centre, School of Accountancy and Business
Nanyang Technological University, Singapore 639798
{p7408153c,ahlchiang}@ntu.edu.sg
[2] Center for Advanced Information Systems, School of Applied Science
Nanyang Technological University, Singapore 639798
aseplim@ntu.edu.sg

Abstract. Many different kinds of algorithms have been developed to discover relationships between two attribute groups (e.g., association rule discovery algorithms, functional dependency discovery algorithms, and correlation tests). Of these algorithms, only the correlation tests discover relationships using the *measurement scales* of attribute groups. Measurement scales determine whether order or distance information should be considered in the relationship discovery process. Order and distance information limits the possible forms a legitimate relationship between two attribute groups can have. Since this information is considered in correlation tests, the relationships discovered tend not to be spurious. Furthermore, the result of a correlation test can be empirically evaluated by measuring its significance. Often, the appropriate correlation test to apply on an attribute group pair must be selected manually, as information required to identify the appropriate test (e.g., the measurement scale of the attribute groups) is not available in the database. However, information required for test identification can be *inferred* from the system catalog, and analysis of the values of the attribute groups. In this paper, we propose a (semi-) automated correlation test identification method which infers information for identifying appropriate tests, and measures the correlation between attribute group pairs.

1 Introduction

Many algorithms have been developed to discover the extent that the values of a pair of attribute groups associate with each other. These *relationship discovery* algorithms include algorithms that mine for functional dependencies [13], association rules [1], and correlations [2].

Most of the recently developed algorithms discover relationships between attribute groups without taking into account information such as the measurement scale, or statistical significance of the results. Thus, the relationships discovered by these algorithms are often spurious or erroneous. For example, the support and confidence framework used by association rules may discover relationships that are contrary to the real world situation [3].

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 29–40, 1999.

One type of information that can assist in relationship discovery is the *measurement scale* of the attribute group. Measurement scales specify the measurement properties (distinctiveness, order, distance, and zero value) in effect on an attribute group. Statisticians recognize four different measurement scales, the nominal, ordinal, interval, and ratio scales [2]. The nominal measurement scale indicates that the values of an attribute group are *distinct*. For example, `Religion` has a nominal measurement scale. All we know about the values 'Catholic', and 'Buddhist' is that they describe different religions. The ordinal measurement scale indicates that the values of an attribute group not only are distinct, but have an *intrinsic order* as well. For example, `School_Ranking` has an ordinal measurement scale. Being the '1st' in school is better than being the '2nd'. The interval measurement scale indicates that values are not only distinct, and have order, but also the distance between the values can be measured. For example, `Date_of_Birth` has an interval measurement scale. Someone born on September 4, 1976 was born 2 days after someone born on September 2, 1976. The ratio measurement scale has all of the properties of the interval measurement scale. In addition, one value of the ratio measurement scale conforms to a 'zero value'. This 'zero value' indicates the absence of the property that the attribute group measures. For example `No_of_Children` is on the ratio scale. If you have 0 children, you have no children.

An example of how measurement scale information is useful in determining the spuriousness of a relationship is given in Table 1. In Table 1, a one-to-one mapping can be established between the values of `Salary`, and `Time_Leave_Home`. This mapping seems to imply that a relationship exists between `Salary`, and `Time_Leave_Home`. However, `Salary`, and `Time_Leave_Home` have the interval measurement scale, and can be treated as continuous. We can therefore expect that a genuine relationship between these two attributes would be a continuous function. However, the Intermediate Value Theorem [16] states that for any continuous function, every value Y between some pair of values B_1 and B_2 will have a corresponding value X between the pair of values A_1, and A_2, where A_1 is associated with B_1, and A_2 is associated with B_2. In Table 1, we see no evidence that the Central Limit Theorem holds. Thus, we can suspect that the relationship between `Salary`, and `Time_Leave_Home` is spurious.

Table 1. An Artificial Relationship

Salary	Time_Leave_Home
10,425.00	7:45
15,492.66	6:30
20,485.42	8:15
15,628.23	7:54
27,154.21	7:05

Correlation tests employ the measurement scale to measure the relationship between two attribute groups. As they exploit the properties of measurement scales, relationships they discover tend to be less spurious than other relationship discovery algorithms which do not exploit this information. This is important,

as large databases may contain many spurious relationships. In addition, the results of correlation tests can also be validated by measuring the *significance* of the results. Significance measures the probability that a relationship discovered by the correlation test is a spurious one [2].

However, there are problems with using correlation tests for data mining. These problems include: 1) the identification of the measurement scale of an attribute group, and 2) the need for more information than just the measurement scales to determine the appropriate correlation test. First, to identify the appropriate correlation test, the measurement scales of the attribute groups must be known. For example, one of the correlation tests for interval measurement scales, such as the coefficient of determination should be used to measure the correlation of an attribute group pair with interval measurement scales. The correlation coefficient of an attribute group pair with nominal measurement scales should be calculated using a correlation test such as the phi coefficient. However, because of the large number of attributes in the relation, the data mining specialist cannot be expected to determine the measurement scale for every single attribute group manually.

Also, additional information needs to be derived in order to determine the appropriate correlation test from those usable for each measurement scale. For example, both the phi coefficient, and the contingency coefficient are used on attribute group pairs with nominal measurement scales. However, the phi coefficient should be employed only when both attribute groups have at most two distinct values, and the contingency coefficient should be applied when any attribute group has three or more distinct values. This information can be inferred from the values, length, and data type of the attribute.

Finally, the problem of identifying correlation tests for relationship discovery in data mining is complicated by the need to identify the appropriate correlation tests not only for the individual attribute pairs, but also the attribute group pairs. For example, while the attributes `Monthly_Salary`, `Bonus`, `Net_Yearly_Salary`, and `Taxes` may have no strong pair-wise relationships, the combination {`Monthly_Salary, Bonus`} may correlate strongly with {`Net_Yearly_Salary, Taxes`}, as the attributes are related to each other through the function `Monthly_Salary` $\times 12+$ `Bonus` = `Net_Yearly_Salary` + `Taxes`.

Therefore, a (semi-)automatic method should be developed to facilitate determining the appropriate correlation tests. In this paper, we propose and discuss a (semi-)automated correlation test identification method. The result of the appropriate correlation test (i.e. correlation coefficient) measures the strength of the relationship of the attribute group pair. While our correlation test identification method assumes that the database to be mined follows the relational database model, it can be generalized to other database models as well. In this paper, the discussion of our method focuses on the attribute groups of a single relation only.

Problem Definition- Let R be a relation with attributes $\mathtt{A}= \{\mathtt{A}_1, \mathtt{A}_2, \ldots, \mathtt{A}_n\}$. $G = 2^A$ is the set of attribute groups. Let $P_{ij} = (g_i, g_j)$ be a relationship between two mutually exclusive attribute groups, i.e. $g_i \in G$, $g_j \in G, g_i \cap g_j = \emptyset\}$. Given

a P_{ij}, calculate the *correlation coefficient* S_{ij} (which determines the strength of the relationship between the attribute groups of the pair). To calculate S_{ij}, we must first identify the appropriate correlation test T for (g_i, g_j).

2 Overview of the Correlation Test Identification Method

Attribute characteristics necessary for determining the appropriate test must be identified. We identify these characteristics and classify them into a set of *representations*. For each attribute, at most one representation should be identified. The representation can be identified by analyzing the instances, data type, and length of each attribute. We assume that the system catalog records the data types of attributes. If the system catalog does not contain information on the length of attributes, this information can be derived from the instances. In addition, the representations of the individual attributes are used to infer the representations of the attribute groups. The two representations of an attribute group pair will then determine the appropriate correlation test used to measure their relationship.

Data instances are used to infer the representation of an attribute. In a large relation, examining all data instances may take too much time. To speed up processing, a sample may be used for analysis as long as that sample is representative. Our method incorporates a formula which estimates the representative sample size.

There are four steps in the correlation test identification method:

Step 1: **Take a Representative Sample**. A representative sample of the relation is extracted to speed up relationship discovery. The size of the sample is determined based on the acceptable degree of error, and the acceptable probability that this degree of error will be exceeded. The formula for calculating the sample size (Step 1) is discussed in Section 3.

Step 2: **Assign Representations to All Attribute Groups**. Each attribute is assigned a representation based on the analysis of its data type, length of the attribute, and values. The representations of attributes are then used to determine the representations of attribute groups. The identification of representations for attributes, and attribute groups (Step 2) are discussed in Section 4, and 5 respectively.

Step 3: **Select Tests**. The attribute groups are paired to form all possible sets of P_{ij}. The representations of the two attribute groups in each pair are then used to identify the most appropriate correlation test. Section 6 discusses the identification of the appropriate correlation tests.

Step 4: **Execute and Evaluate the Correlation Test Result**. The correlation test is executed, and the result and its significance is analyzed. If the result of the test is strong and significant, it is validated by executing the correlation test on other samples in the same attribute group pair. The evaluation of correlation tests is discussed in Section 7.

3 Identifying a Representative Sample

Most of the time, the relation to be mined will contain a large number of instances. Any examination of all instances in the relation would take a very long time. If a random sample of the data in the relation is extracted, it often will be representative of the original data set. Thus, while mining the *representative* sample is quicker than mining the relation, the result of mining the sample will often be comparable to that of mining the relation.

Our method uses the worst case of the formula for calculating sample size from estimated proportions $n = \frac{p \times (1-p) \times Z(\alpha/2)^2}{\epsilon^2}$ to generate our sample size, where p is a value between 0 and 0.5 which indicates a degree of variability, n is the size of the sample, ϵ is a degree of error, α is a probability this degree of error will be exceeded, and Z is a function describing the area under the standard normal curve [10]. As we do not know the degree of variability, we set p to be the worst case value of 0.5. This formula can be used to estimate sample size in any relation with a large number of instances, as it assumes that the number of instances is *infinite*.

4 Assigning Representations to Individual Attributes

4.1 Representations

To identify the appropriate correlation test for an attribute group pair, we need to know more than just the measurement scale. The measurement scales are subdivided into a set of *representations*, which captures this additional information.

We are not aware of any correlation test which exploits zero value information to measure a relationship. Therefore, we do not differentiate between attribute groups with the ratio and interval measurement scales. For the interval measurement scale, two major characteristics of the data play a major part in determining the appropriate correlation test. First, the various tests for attributes groups with interval measurement scales can only handle attribute groups with certain numbers of attributes. For example, the coefficient of determination requires that of the two attribute groups being compared, one of them contains only one attribute. Second, dates differ from other kinds of attributes with the interval measurement scale, because multiplication, division, exponentiation, etc. cannot be performed on dates. Thus, two date attribute groups $\{X_1, X_2, \dots\}, \{Y_1, Y_2, \dots\}$ can relate to each other only through a linear functional form, i.e. $X_1 \pm X_2 \pm \ldots \pm C = Y_1 \pm Y_2 \pm \ldots$.

Few correlation tests exist for the ordinal measurement scale, so it is not necessary to further partition it. However, for the nominal measurement scale, special case tests exist which exploit the special characteristics of *dichotomies*. Dichotomies are attribute groups with nominal measurement scales which have only two distinct values. Since dichotomies only have two values, they have characteristics that are different from other nominal attribute groups. For example, it can be assumed that all values in the dichotomy are equidistant. In the dichotomy

`Sex` with values {M, F}, we can say that |M − F| = |F − M|. However, for a non-dichotomy like `Religion`, it cannot be assumed that |Catholic − Buddhist| = |Buddhist − Moslem|. We subdivide the nominal measurement scale into two scales, one scale for dichotomies, and one scale for non-dichotomies.

We take these characteristics of the measurement scales into account by partitioning the measurement scales into the following representations:

- Multi-attribute Numeric (MAN)- An attribute group with this representation has an interval measurement scale. This attribute group contains more than one attribute.
- Single-attribute Numeric (SAN)- An attribute group with this representation has an interval measurement scale. This attribute group contains only one attribute.
- Single-attribute Date (SAD)-This is an attribute group which represents temporal data (e.g., The attribute group {`Date_of_Birth`} is a SAD). This attribute group contains only one attribute.
- Multi-attribute Date (MAD)- An attribute group with this representation uses several attributes to describe a temporal 'fact'.
- Ordinal(Ord.)- The Ordinal representation indicates that while the values of the attribute group have a ranking order, no information is available to determine the *distance* between the values.
- Categorical(Cat.)- This representation means that the only measurement property found in the attribute group is distinctness.
- Dichotomous(Dich.)-Attribute groups with the Dichotomous representation can only contain two distinct values, e.g., {M, F}, {0, 1} etc.

The measurement scales are partitioned into the following representations: 1) The Interval measurement scale is partitioned into the MAN, SAN, MAD, and SAD representations, 2) The Ordinal measurement scale has the analogous Ordinal representation, and 3) The Nominal measurement scale is partitioned into the Categorical and Dichotomous representations. In this paper, any discussion of a measurement scale applies to all representations with that measurement scale. In addition, when we refer to a Numeric representation, we mean the MAN, and SAN representations collectively. When we refer to a Date representation, we mean the SAD and MAD representations collectively.

4.2 Data Types

The data type of an attribute is useful in determining its representation. However, different RDBMSes use different labels to describe the same data types. For the purpose of this paper, we assume that the following are the data types available in the RDBMS:

- Integers- The values of attributes with this data type differ from each other in increments of at least one unit. For example, the attribute `Years_Of_Service` is often given an Integer data type.

- Decimals - Attributes with this data type may have values which differ from each other by less than one unit. For example, the attribute `Salary` can be given the Decimal data type.
- Date- A data type where user input is restricted to specifying a day, month, and year. For example, `Date_Of_Birth` is often assigned a Date data type.
- String-Each character in this data type may have any value.

4.3 Identifying and Eliminating Representations for Single Attributes

An attribute with a particular data type can only have certain representations. The possible representations of each data type are shown in Table 2. A set of heuristic rules are then applied to the attribute to identify which of the possible representations is the correct one. Many of these heuristic rules have been adapted from rules used in SNOUT [15] to identify measurement scales from survey data.

Table 2. Initially Generated Hypotheses

Type	SAN	SAD	ORDINAL	CATEGORICAL	DICHOTOMOUS
Integer	√	√	√	√	√
Decimal	√	√			
String	√	√	√	√	√
Date		√			

The heuristic rules are listed below. Each heuristic rule uses more information to determine the correct representation than the previous rule. However, the additional information used by each rule is less reliable as compared to information added by the previous rule. Thus, representations identified by an earlier rule can be said to be more accurate than representations identified by a later rule.

1. Attributes with the Date data type have the DATE representation.
2. If the attribute has a possible DICHOTOMOUS representation, and the number of distinct values of the attribute is two or less, the attribute has the DICHOTOMOUS representation. If the number of values is more than two, the attribute can not have the DICHOTOMOUS representation.
3. If the length of the attribute varies across the values, it cannot have the SAD representation.
4. If the values of an attribute with the String data type contains non-numeric characters, then the attribute cannot have the SAN representation.
5. If an attribute does not conform to an accepted date format, it cannot have the SAD representation. For example, an attribute with the instance '231297' might have the date representation, since this instance could mean December 23, 1997. However, if the same attribute had another instance '122397', it could not have the date representation, since both instances could not indicate a date under the same format.

6. If an attribute is longer than nine characters, it cannot have the ORDINAL or CATEGORICAL representations. If an attribute has more than 25 distinct values, it cannot have these two representations either. An attribute with less than 25 distinct values cannot have the SAN representation [15].
7. Attributes with the Integer data type can not have the CATEGORICAL representation if the length of the values is greater than 2. If the difference between the minimum and maximum value of attributes with the Integer data type is greater than 25, it will not have the CATEGORICAL REPRESENTATION. This rule follows as a direct consequence of rule 6. If an attribute with the String data type has only values with characters between '0' and '9', then it must also follow this rule.
8. If the first character of any value of an attribute with the String data type differs from the first character of all other values by 3 or more, (e.g., 'D' differs from 'A' by 3), then the attribute does not have an ORDINAL representation.
9. If an attribute may have both a SAD and a SAN representation, the attribute will not have the SAN representation.
   Justification: The range of values which can indicate a SAD forms only a small proportion of the range of values which can indicate a SAN. If every single value in an attribute falls into the range of values that indicate a SAD, the attribute is more likely to represent a SAD than a SAN.
10. If an attribute can have both an ORDINAL and a CATEGORICAL representation, the attribute cannot have the ORDINAL representation.
    Justification: It is often difficult to identify whether an attribute really has an ORDINAL or a CATEGORICAL representation based on the schema and instance information alone. However, a correlation test designed for attribute groups with CATEGORICAL representations can be correctly used to compare attributes with ORDINAL representations. However, the reverse is not true. By using this rule, we err on the side of caution.
11. Attributes which may have SAD, and ORDINAL, or SAD, and CATEGORICAL representations have the SAD representation. The reasoning in this rule is similar to that of rule 9.

After an attribute has been processed using these rules, it is possible for it to have no representation. This indicates that the attribute can not be analyzed using correlation tests. After the system has discovered the representations for the individual attributes, the data mining specialist may review the results, and change any representation he or she deems incorrect.

5 Identifying Representations for Attribute Groups

The representation for an attribute group which contains only one attribute is the same as the representation of that attribute. The representation for an attribute group containing more than one attribute is determined by consulting Table 3. We discuss some of the counterintuitive derivations of attribute group representations in this section.

Table 3. Representations of Attribute Groups With More Than One Attribute

Attr. Grp 1 \ Attr. Grp 2	MAN	SAN	MAD	SAD	ORD.	CAT.	DICH.
MAN	MAN	MAN	MAD	MAD	N/A	N/A	N/A
SAN	MAN	MAN	MAD	MAD	N/A	N/A	N/A
MAD	MAD	MAD	MAD	MAD	N/A	N/A	N/A
SAD	MAD	MAD	MAD	MAD	N/A	N/A	N/A
ORD.	N/A	N/A	N/A	N/A	CAT.	CAT.	CAT.
CAT.	N/A	N/A	N/A	N/A	CAT.	CAT.	CAT.
DICH.	N/A	N/A	N/A	N/A	CAT.	CAT.	CAT.

An attribute group with the NUMERIC *representation combined with an attribute group with the* DATE *representation produces an attribute group with the* MAD *representation.* This rule reflects the situations when numbers are used to increment or decrement a date. For example, when the attribute group with SAN representation {`Project_Duration`} is combined with the attribute group with SAD representation {`Date_Started`} it produces the attribute group {`Project_Duration, Date_Started`}. The combined attribute group identifies the date the project was completed.

An attribute group with the ORDINAL *representation combined with an attribute group with the* ORDINAL *representation produces an attribute group with the* CATEGORICAL *representation:* Intuitively, the combination of two attribute groups with ORDINAL representations should result in an attribute group with the ORDINAL representation. We do not allow this for two reasons. First, the ordering priority of the two attribute groups is often not self evident. For example, the attribute group {A_1, A_2} may be ordered as (A_1, A_2) or as (A_2, A_1). Second, we have found no correlation test which allows attribute groups with ORDINAL representations to have more than one attribute. Instead, we downgrade the representation of the combined attribute group to a CATEGORICAL representation. Similarly, an attribute group with an ORDINAL representation is treated as if it had a CATEGORICAL representation when it is combined with attribute groups having INTERVAL representations.

An attribute group with the INTERVAL *representation and an attribute group with the* NOMINAL *representation cannot be automatically combined.* For the two attribute groups to be combined, the attribute group with an INTERVAL representation would have to have its values transformed to values which can be compared using a correlation test for CATEGORICAL, or DICHOTOMOUS representations. It is not possible to perform this transformation automatically.

It is not possible to convert the values of an attribute group with the INTERVAL representation to values acceptable for the DICHOTOMOUS representation, as there are too many distinct values in an attribute group with the INTERVAL representation. While a sample of the instances of an attribute group with the INTERVAL representation does not physically capture all the distinct values, missing values can still be extrapolated from the values present in the sample.

For example, while the sample may not have the value 49, the presence of 49 may still be inferred because the values 50 and 48 exist in the sample. "Dichotomization" destroys the notion of distance between values, and thus the inferred values are lost in the process. This would make the sample unrepresentative.

The conversion of an attribute with the INTERVAL representation to one with the CATEGORICAL representation cannot be automatically performed, as the appropriate categorization system, and the appropriate number of categories are often not apparent from an examination of the data. Arbitrary selection of a categorization method may lead to incorrect categorization. If categorization is performed manually, the combined representation of an attribute group with the INTERVAL representation, and one with the CATEGORICAL representation will be CATEGORICAL.

6 Measuring Relationships between Attribute Groups

Once representations for all attribute groups have been identified, all mutually exclusive attribute groups can then be paired for analysis. Depending on the representation of each attribute group in the pair, one of the tests in Table 4 is selected to measure the correlation of the attribute groups.

Table 4. Tests to Compare Relationships Among Attribute Groups

Repr.	MAN	SAN	MAD	SAD	ORDINAL	CATEGORICAL	DICHOTOMOUS
MAN	Canon. Corr.	Box-Tidwell	Canon. Corr.	Box-Tidwell	MANOVA	MANOVA	Log. Regr.
SAN	Box-Tidwell	Box-Cox	Box-Tidwell	Box-Cox	Spearman	ANOVA	Pt. Biserial
MAD	Canon. Corr.	Box-Tidwell	Canon. Corr.	Pearson Corr.	MANOVA	MANOVA	Log. Regr.
SAD	Box-Tidwell	Box-Cox	Pearson Corr.	Pearson Corr.	Spearman	ANOVA	Pt. Biserial
ORD	MANOVA	Spearman	MANOVA	Spearman	Spearman	Cntgcy. Coeff.	Ordered Log.
CAT	MANOVA	ANOVA	MANOVA	ANOVA	Cntgcy. Coeff.	Cntgcy. Coeff.	Cntgcy. Coeff.
DICH	Log. Regr.	Pt. Biserial	Log. Regr.	Pt. Biserial	Ordered Log.	Cntgcy. Coeff.	Phi Coeff.

Most of the tests described in Table 4 are common and accepted tests for comparing attribute groups with the representations described. They are discussed in most classic statistics textbooks (e.g., [2,9,14]).

7 Evaluating the Correlation Tests

Once an appropriate correlation test for a pair of attribute groups has been determined, the test is performed against the representative sample extracted from the instances of the attribute group pair. The test will yield two values, the correlation coefficient, and the statistical significance. The correlation coefficient measures the degree to which the values of one attribute group predict the values of another. The value of the ANOVA which is comparable to the correlation coefficient is the *F-ratio*, which measures the difference in the distribution of interval values associated with each categorical value.

The *statistical significance* indicates the probability that the relationship being discovered occurred as a result of chance. The smaller the significance value, the more certain we are that the correlation test found a genuine relationship.

A maximum significance threshold, and minimum correlation coefficient threshold for each test can be specified by the mining specialist prior to relationship discovery. Only attribute group pairs which have correlation coefficient values above the coefficient threshold, and significance values below the significance threshold will then be confirmed.

Like other kinds of knowledge discovered by data mining, relationships discovered by correlation tests should not be taken as gospel until they are verified and validated. While setting the significance threshold to a low value would reduce the number of discovered relationships that are spurious, some spurious relationships will still be discovered. Furthermore, setting the significance threshold to an extremely low value will cause the method to reject many genuine relationships, thus rendering the method less effective.

The results of correlation tests can be validated empirically in several ways. A quick, and reliable way would be to first re-sample values from the attribute group pair identified as having a strong correlation, and then run the same correlation test again. If repeated tests on different samples produce high correlation coefficients, and low significance values, then we are more certain that a genuine relationship exists between the two attribute groups. Of course, if time permits, the best validation would be to perform the correlation test on *all* the values of the attribute group pair.

8 Conclusion

In this paper, we propose and discuss a heuristic method for identifying correlation tests to measure the relationship between attribute group pairs. We have also discussed how correlation tests can provide information not only on the strength, but also the significance of a relationship. We are currently attempting to extend our research in several directions.

First, are in the process of standardizing the results of the various correlation tests. The possible results of the various correlation tests vary widely. For example, the Spearman's Rho, Box-Tidwell, and Canonical Correlation tests have scores ranging from -1 to 1. The minimum score of the contingency coefficient is 0, but the maximum score varies according to the sample size. The result of the F-ratio has a minimum value of 1, and a theoretically infinite maximum. We cannot expect that an untrained user will be able to interpret these varied scores.

Second, we are attempting to apply this method to the database integration problem. In developing this method, we have noted the similarity between data mining, and the attribute identification problem in database integration. Attribute identification is the sub-problem of database integration which not only deals with identifying equivalent attributes (i.e. attribute equivalence [12]), but sets of attributes as well. Finding representations for attribute identification is a problem of significantly larger scope than finding representations for data mining, since relationships between attribute groups with String and Key representations must also be accounted for. We are currently investigating the

applicability of an extended version of our correlation test identification method to the attribute identification problem.

Finally, we are looking for ways to validate our method. As it is difficult to mathematically validate heuristic methods, we are attempting to validate the heuristics employed against real world databases. Currently, we are acquiring a large variety of data sets to validate our method against. Results from preliminary tests on small, publicly available data sets (e.g., [4,8]) are encouraging. However, further tests still need to be performed.

References

1. R. Agrawal, T. Imielinski, A. Swami. *Mining Association Rules Between Sets of Items in Large Databases.* Proc. of the ACM SIGMOD Conference on Management of Data. pp. 207-216. 29
2. R.B. Burns. *Introduction to Research Methods - Third Edition.* Addison-Wesley. 1997. 29, 30, 31, 38
3. S. Brin, R. Motwani, C. Silverstein. *Beyond Market Baskets: Generalizing Association Rules to Correlations.* Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. 1997. pp. 265-276. 29
4. T.J. Biblarz, A.E. Raftery. *The Effects of Family Disruption on Social Mobility.* American Sociological Review. 1993. 40
5. B. Everitt. *Cluster Analysis.* Heinemann Educational Books. 1980.
6. J.E. Freund, R.E. Walpole. *Mathematical Statistics- Fourth Edition.* Prentice-Hall. 1987.
7. J.D. Gibbons. *Nonparametric Methods for Quantitative Analysis (Second Edition).* American Sciences Press Inc. 1985.
8. V. Greaney, and T. Kelleghan. *Equality of Opportunity in Irish Schools.* Dublin: Educational Company. 1984. 40
9. J.F. Hair Jr., R.E. Anderson, R.L. Tatham, and W.C. Black. *Multivariate Data Analysis with Readings.* Prentice Hall. 1995. 38
10. D.V. Huntsberger, and P.P. Billingsley. *Elements of Statistical Inference.* Allyn and Bacon Inc. 1987. 33
11. J.S. Long. *Regression Models for Categorical and Limited Dependent Variables.* Sage Publications. 1997.
12. J.A. Larson, S.B. Navathe, R. Elmasri. *A Theory of Attribute Equivalence in Databases with Application to Schema Integration.* IEEE Transactions on Software Engineering. April 1989. pp. 449-463. 39
13. H. Mannila, and K.J. Raiha. *Algorithms for Inferring Functional Dependencies From Relations.* Data and Knowledge Engineering. February, 1994. pp. 83-90. 29
14. J. Neter, W. Wasserman, M.H. Kutner. *Applied Linear Regression Models. 2nd Edition.* Irwin Homewood. 1989. 38
15. P.D. Scott, A.P.M. Coxon, M.H. Hobbs, R.J. Williams. *SNOUT: An Intelligent Assistant for Exploratory Data Analysis.* Principles of Knowledge Discovery and Data Mining. 1997. pp. 189-199. 35, 36
16. G.B. Thomas, and R.L. Finney. *Calculus and Analytic Geometry.* Addison-Wesley. 1996. 30

Incremental Meta-Mining from Large Temporal Data Sets[1]

Tamas Abraham and John F. Roddick

Advanced Computing Research Centre
School of Computer and Information Science,
University of South Australia
The Levels Campus, Mawson Lakes, Adelaide,
South Australia 5095
{abraham, roddick}@cis.unisa.edu.au

Abstract. With the increase in the size of datasets, data mining has become one of the most prevalent topics for research in database systems. The output from this process, the generation of rules of various types, has raised the question of how rules can be considered interesting. We argue that, in many cases, it is the meta-rule that holds the most interest. That is, given a set of known rules about a dataset, it is the confluence of rules relating to a small subset of characteristics that commonly becomes the focus of interest. In earlier work, we investigated the manner in which meta-rules, rules describing rules, could be discovered and used within a data mining system. In this paper we extend this and present an approach that enable meta-rules to be found incrementally. The emphasis of the work is on temporal data mining as we find that temporal data readily lends itself to data mining techniques, however, as can be seen from the paper, the temporal component can easily be abstracted out and the results are thus also applicable in a non-temporal domain.

1 Introduction

Data mining has recently become a common research topic in the field of database systems partly due to the increase in interest in size of datasets, and partly due to the maturity of current database and machine learning technology. The output from this process, the generation of rules of various types, has raised the fundamental question of how the generation (or at least the presentation) of a potentially limitless number of rules can be restricted to those that are in some way *interesting*. We argue that, in many cases, it is the meta-rule that holds the most interest. That is, given a set of known rules about a dataset, it is the confluence of rules relating to a small subset of characteristics that commonly becomes the focus of interest. In earlier work (Abraham and Roddick 1997a) we investigated the manner in which meta-rules - rules describing rules, could be discovered and used within a data mining system. In this paper we extend this and present an approach that enable meta-rules to be found incrementally. In this respect, it parallels the incremental data mining work done on normal data although some interesting differences have been found and are discussed.

[1] This paper is an abstract of a longer Technical Report CIS-98-010 available from the School of Computer and Information Science at the address above.

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 41-55, 1999.

For many reasons, the emphasis of much of our work (q.v. Rainsford and Roddick 1996; Abraham and Roddick 1997b, 1998b) relates to temporal data mining as we find that temporal data readily lends itself to non-trivial data mining techniques that commonly suggest cause and effect – one of the most useful, yet most difficult connections to make manually. However, as can be seen from the paper, the temporal component can easily be abstracted out and the results are thus also applicable in a non-temporal domain. Our framework includes the following:

- Definitions of the elements within the framework (in this case, meta-rules and other supporting components).
- Naming conditions that enable the meta-rule mining process to take place (the environment and the existence of pre-compiled rule sets).
- Describing the process itself.

Throughout this paper, we will refer to meta-rules as rules that express changes in two or more rule sets possibly compiled by different methods at different times. This concept of examining changes in evolving rule sets is dissimilar to previous efforts such as rule maintenance (Cheung, *et al.* 1996; Cheung, Ng and Tam 1996), which generally do not retain earlier versions of an updated rule, and thus no comparison between new and old versions is possible. The term meta-rule has been chosen to describe the fact that we are constructing rules about already existing rules (or even meta-rules), by way of describing the variation of patterns between rule set layers.

The paper is sub-divided as follows. The remainder of the introduction introduces meta-rules and discusses the meta-rule set base and the definition of difference measures. Section 2 discusses the processing of the meta-rule set base and the meta-rule mining process itself. This is a synopsis of the work formerly presented in (Abraham and Roddick 1997a) and is included here (and in more detail in (Abraham and Roddick 1998a)) for completeness. Section 3 looks at the applications of meta-rules and meta-rule mining in a number of different application domains while Section 4 presents a framework for mining evolution patterns in spatio-temporal data.

1.1 Meta-Rules and the Meta-Rule Set Base

Many of the terms used in data mining and knowledge discovery literature have many meanings and a number of terms need to be clarified.

Definition 1. A *rule r* in rule set R on data set D describes a characterisation of the contents of D in an *If A then B with caveat c* format, where A and B are concepts/patterns represented or implied in D and caveat c may be a probability or some other kind of expression indicating rule feasibility. A *meta-rule m* in meta-rule set M on rule set R then describes a characterisation of the contents of R in the same fashion.

As this definition is very general (and can be extended to define meta-meta-rules etc.), we give an interpretation of it for the snapshot data case. By allowing the data

set D to be a collection of snapshots $\{D_1, ..., D_n\}$ over the same domain with their corresponding homogenous rule sets compiled into $R = \{R_1, ..., R_n\}$, R_i being the rule set extracted for D_i, $i = 1,... , n$, a meta-rule set $M_{i,j}$ relates to rules belonging to rule set layer pairs (R_i, R_j) of R and will express differences between the two rule set layers forming the pair. The collection of all $M_{i,j}$, $i, j = 1,... , n$, makes up the meta-rule set M over R. This can be viewed as a restriction of the original definition of meta-rules as characterisation (in this case, difference description) is only performed for pairs of selected subsets and not for R as a whole, but it is a convenient and capable way to handle the snapshot nature of the original data.

Definition 2. Given two rule sets R_1 and R_2 of the same rule type, compiled on the same data domain but registered at different times, with set R_2 being for the later time, we define four categories of rule to make up the *meta-rule set base* between R_1 and R_2:

- New Rules. Rules in rule set R_2 that do not themselves or, in some variant (modified) form, appear in R_1.
- Expired Rules. Rules in rule set R_1 that do not themselves or, in a variant form, appear in R_2.
- Unchanged Rules. Rules that appear in both R_1 and R_2 in the same form.
- Changed Rules. Rules that appear in both R_1 and R_2, but in a different form.

Depending on the rule type, the extent of modification necessary to change an existing rule is determined by some predefined difference measure.

2 Processing the Meta-Rule Set Base and the Meta-Mining Process

The new, expired, unchanged and changed rule categories can be obtained by an appropriate separation algorithm and are presented in the same normalised format as the input rule sets, except for the changed category. This happens because the change in a given rule must be explicitly recorded to facilitate later processing. Therefore, rules in the changed category of the meta-rule set base will have an *extended* normalised format, recording both the old and new values of the designated difference measures. For example, for association rules, two new columns are added to the original four to record the antecedent, consequent, old and new confidence and support values. Alternatively, if the interest measures are quantitative and we are only interested in the deviation in the difference measures, the magnitudal change from the original values can be stored instead of new values (or may be omitted altogether). An *altered* normalised form would be used in such cases instead of an extended one. In the association example, columns to be used form a set containing the {antecedent, consequent, confidence and support percentage *deviation*}. Note that this differs from the original format of {antecedent, consequent, confidence and support} in that the confidence and support deviations can be negative as well as positive, while the original confidence and support values are always non-negative.

In processing the categorised meta-rule set base, we mentioned general descriptions, or generalisation as the obvious choice. Alternatives may exist (such as mining associations within each group), but we shall concentrate on the individual generalisation of the four categories in further discussions. Three of these categories (new, expired and unchanged) are in the original normalised rule input format. Processing them can therefore be done in the same fashion. Because the rules are normalised, the contents of each category can be treated as the data source for the particular mining algorithm employed. In addition, if the original rules were derived using some kind of background knowledge, typically conceptual hierarchies, then they can be re-utilised by the algorithm, e.g. the previously mentioned attribute-oriented induction technique of Han, *et al.* (Han, Cai and Cercone 1992). Thus, higher level conceptualisations of the rules in each category can be attained for both the constant rule parts and difference measures. Of course, if no conceptual hierarchies were available at the time of generating the original rule sets, then they may need to be provided before induction can take place.

In the case of the changed rules category, we may employ the same above strategy when extracting general descriptions for the constant rule parts. On the other hand, to measure and generalise change in the difference measures, separate *change magnitude hierarchies* must be introduced. This, in fact, is the area where the most powerful observations can be made.

Whilst meta-rules are capable of expressing the general characteristics of the meta-rule set base categories, some obvious questions remained unanswered in previous sections. Three of these issues are listed below and are discussed in more detail in (Abraham and Roddick 1998).

- **Identification**. A meta-rule, as stated above, relates to rules in different rule sets, or more strictly, different layers of a given rule set. Therefore, it must be identifiable with these rule set layers.

- **Existence**. The four categories in the meta-rule set base may or may not exist for every rule type. For example, spatial dominant generalisation of geo-referenced data, (qv. Lu, Han and Ooi 1993) does not have new or expired rules as rules in both sets relate to the same generalised spatial areas determined by the generalisation thresholds.

- **Feasibility**. In the case of some rule types we may not require the use of meta-rules if the rule sets are small and comparison is easy by other means. A typical example could be the extraction of general descriptions for only a limited number of spatial areas. Association rule sets, on the other hand, are often quite large (Klemettinen, *et al.* 1994), and are therefore one of the possible areas where meta-rule discovery can be most beneficial.

2.1 The Meta-Rule Mining Process

Meta-rule mining is only one component of the process. The typical meta-rule extraction process consists of four steps (Abraham and Roddick 1997a).

- **Rule set generation**. In this step, we select and mine individual snapshots and collect the associated rules into separate rule sets (timestamped for identification). The rules that are generated are commonly of the same rule type, and the same hierarchies for background knowledge, also including possible generalisation thresholds, etc., must be used to ensure full compatibility between the resulting rule set layers.
- **Input preparation**. This is the stage of preparation before meta-rule mining. If necessary, we convert the contents of the snapshot rule set layers into a consistent format to facilitate rule processing.
- **Meta-rule set base generation**. A separation algorithm that takes two rule set layers as input, compares them and produces the four categories of the meta-rule set base: *new*, *retired*, *unchanged* and *changed* rules, some of which may be null.
- **Processing of categories**. The meta-rule mining process is concluded by individually processing each meta-rule set base category to derive general characterisations for the contents of each of the four (or possibly only some selected, e.g. *new* and *expired*) categories. This step is the one that constitutes the second level of abstraction described by the term *meta-rule*.

3 Applying Meta-Rules in Different Domains

In the previous section we concentrated on extracting meta-rules from a number of rule sets compiled on time-stamped data layers. In this section, we demonstrate this process by providing a specific example and discuss modifications to the original concept to accommodate the incremental update of the meta-rule set as the data eventually gets refreshed.

We shall use and extend the example used in (Lu, Han and Ooi 1993), in which temperature data is generalised for regions of British Columbia, Canada. Spatially distributed data collection points are allocated to regions that are obtained by conceptual hierarchy ascension and generalisation thresholds.

Area	*Temperature*
Mid-Central	Hot
…	…
North-Central	Mild
North-Central	Mild
North-Central	Mild
…	…
South-West	Moderate

Table 1 : Generalised region temperatures

For each region, temperature data values are averaged and generalised to higher-level temperature concepts. The resulting table contains (*region, temperature*) pairs with values such as (North-West, mild) and (Mid-Central, hot), see Table 1. Thus, stage 1 of the meta-rule generation process starts with two spatial-data dominant generalisations of the same spatial area, using the same conceptual hierarchies and generalisation thresholds, but for temperature attribute data registered at two different times.

Stage 2 of the meta-rule mining process, input preparation, can be omitted as the rules are expressed in a two-column tabular format by default.

Area	*Temperature*
North-Central	Mild
North-Central	Mild
North-Central	Mild

Area	*Old Temperature*	*New Temperature*
Mid-Central	Hot	Warm
Mid-East	Warm	Moderate
Mid-West	Moderate	Warm
South Central	Mild	Cool
South-East	Warm	Cool
South-West	Moderate	Cool

Table 2: Unchanged and changed generalised temperatures: An example

The next step (Stage 3) is to use a merge-and-split algorithm with the region description being the constant part of the rules and the temperature concept the difference measure. Because we required the use of the same conceptual hierarchies and generalisation thresholds for the regions, this ensures that they remain the same for both data layers, ie. there is a one-to-one correspondence between regions in the two tables. Therefore, there will not be any new and expired rule categories. The unchanged category contains (*region, temperature*) pairs as in Table 1 that have the same temperature concept in both sets, e.g. (Mid-West, mild), while the changed category holds (*region, old temperature, new temperature*) triplets such as (Mid-East, hot, moderately hot), see Table 2, with the total number of entries in the two groups equalling the number of existing generalised regions.

The final step in the meta-rule mining process describes further the contents of these two categories. In the unchanged case, climbing the spatial conceptual hierarchy may merge adjoining regions. This may be done even if the temperatures are different for the merged regions. In this case, the resulting meta-rule may state '*The temperature of the North region remained the same between the surveys*'. In the database, this can be represented by a (*region, temperature set*) pair such as (North, {mild, moderately cold}). Notice that the constancy of the temperature is implied by

the membership in the appropriate meta-rule set base category and does not have to be explicitly stated. Furthermore, if the temperature set contains a single entry, the resulting meta-rule can be even more specific, '*The temperature in the North remained mild*'. Similar results can be achieved for the regions in the changed category. Merging them may immediately produce descriptions such as '*The temperature has changed in the South*'. However, investigating the direction of change may further enhance this. For this purpose, rules may be applied to the (*old temperature, new temperature*) pairs, replacing them with concepts such as 'moderate increase', 'steep decline', and so on, that can be associated with each region producing rules like '*The Mid-East experienced a moderate temperature decrease*', in the case of the above example triplet. Furthermore, regions may be merged using a set of change concepts in a similar fashion to the unchanged case, by applying a restriction on the direction of the change. This may produce a database entry such as (South, {moderate decrease, steep decrease}) that can be interpreted as '*The temperature decreased in the South*', after the change values are further generalised.

3.1 Incremental Maintenance

Until now, we have discussed generating meta-rules for snapshot data. The database used for this purpose does not, however, need to use the snapshot model: snapshots can be taken of continuously changing databases at certain time points and mined. In some circumstances, only the rules generated need to be preserved for meta-rule mining and there may be no need to store the earlier versions of the database itself.

Certain database types (transaction databases, for example) only allow a limited set of operations such as insertion of new records or the reversal of existing records. This makes it relatively easy to keep track of modifications within the database. Clearly, as their contents change, so may the rules that are implied by the data. In order to adjust the rules to reflect the database contents at any given time, the rule generation algorithm could simply be run again. This, however, presents a number of questions. Firstly, when do we run the rule generation algorithm again? Should it be done after each new insertion, or should it wait until a certain threshold is reached? Secondly, is it necessary to re-run the algorithm on the whole data set when only a relatively small number of records may have been inserted (at least, a small number compared to the total number of records)? These questions are addressed by Cheung, *et al.* (1996). Their *rule maintenance* algorithm focuses on the contents of newly inserted data and their effects on already existing rules and reduces processing requirements dramatically.

Another example can be found in Rainsford, *et al.* (Rainsford, Mohania and Roddick 1996), which successfully manages to apply rule maintenance to another data mining technique, in this case, classification. Subsequent work concentrates on enhancing existing algorithms, such as Cheung, Ng and Tam. (1996), to handle multi-level associations, or Cheung, Lee and Kao (1997) to allow maintaining association

rules with more versatile changes permitted in the database, such as modifications or deletions. In addition to describing incremental mining algorithms, the question of when it is appropriate to initiate the maintenance process has received attention (Lee and Cheung 1997).

We now discuss the benefits of maintaining meta-rules. The points to be investigated include:

- The possibility of utilising some of the earlier results or develop new techniques to maintain meta-rules;
- The possibility of maintaining meta-rules – at which level will this happen? Is the maintenance restricted to the incremental updates of the underlying rules or can meta-rules be also incrementally maintained?
- If it is possible to maintain meta-rules, is it viable and/or useful to use this method? Can we show that maintaining meta-rules is more efficient than the direct extraction and generation of rules and meta-rules?
- Finally, if all of the above are possible, can an efficient algorithm to perform incremental maintenance of meta-rules be derived?

To answer these questions, we separate our investigation into two parts: the examination of maintaining the rule sets used as input in the meta mining process and the inspection of possible methods to use to maintain the meta-rules themselves.

Initially, it appears that the use of incremental updates for the input rule sets is indeed an option: because snapshots are taken of the same database, only the difference between these snapshots determines the effectiveness of incremental algorithms. This has been extensively studied for association rules in transaction databases with deletions and insertions allowed by Cheung, *et al.* (Cheung, Lee and Kao 1997) (a modification of an existing record can be treated as a deletion followed by an insertion). They have shown that as long as the changes between the data sets do not exceed a certain limit then incremental updating works faster than re-running existing extraction algorithms. This means, that if appropriate measures are installed to trace the number of updates between snapshots, it becomes possible to select the most efficient algorithm to generate the next rule set, be that either an incremental updating technique or re-running the original extraction algorithm.

Figure 1 contrasts changes that occur in data and rules from one time instance to another. The superscript E (expired) has been used to denote data/rules that exist is the earlier sets but not in the later ones. Similarly, the superscript N (new) has been used to denote data/rules that exist in the later sets but not in the earlier ones.

The main difference that can be observed between the data and rule sets for times T_1 and T_2 in the figure is that we must explicitly mark the rules that changed over time. This can be avoided for raw data as any modification can be treated as a deletion followed by an addition. We cannot, however, allow the same for rules, because one of the major benefits of meta-level discoveries lies in inferring regularities of change. Thus, one of the additional tasks of incremental meta-rule maintenance should be the handling of changed rules from one rule set to another.

The aim of maintaining rules from dynamically changing data is to bring the rules describing the data up to date. The database is constantly updated while the rules are extracted only when this is considered to be important, such as significant change in the data or an upcoming scientific analysis. This means that there is normally some difference between the current contents of the rule set and those that are implied within the present data, unless there have not been any changes to the database since the last rule set generation/update. Meta-rules, on the other hand, describe events in the environment for selected time periods, ie. there is a *from* rule set and a *to* rule set, the two sets used to generate one set of meta-discoveries. In other words, a discovery is tied to a certain period of time and when we maintain meta-rules, we need to consider what happens to both the starting and ending rule set of the period to be updated. Meta-rules will always, at least implicitly, reference the two rule sets that are used to generate them, in order to provide the time frame in which the rules are valid. There is a possible exception to this, though, namely those meta-rules that are valid *now*, ie. are extracted between an old and the current rule set, because these latter observations can be viewed as still in the process.

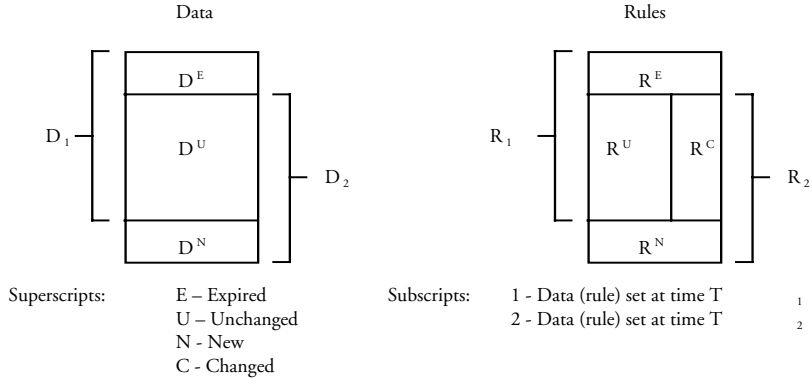


Figure 1: Changing data and rule sets

Figure 2 illustrates some example rule sets and the meta-rule sets that can be generated upon them. Suppose there are some previously compiled rule sets, and a new one is generated. New sets of meta-rules can be generated between this new rule set and any of the previous ones. If there are n previous sets and we add a new one, n new meta-rule sets $M_{i(n+1)}$, $i = 1, ..., n$ can be extracted in all. The following two scenarios exist for creating these meta-rule sets:

a). We extract the meta-rule set for the new and immediately preceding rule sets, ie. $M_{n(n+1)}$. No previous meta information is available for this process, meaning that no incremental procedures can be employed at the meta-level and full extraction needs to be performed. In this kind of meta-rule mining, the advantage gained can only be in the production of R_{n+1} from R_n, by using incremental rule maintenance.

b). We extract the meta-rule set for the new and older rule sets, $M_{i(n+1)}$, $1 \leq i < n$. In any of these cases, a meta-rule set can already exist between the starting rule set R_i and the old ending rule set R_n. Therefore, it may become possible to use the contents of M_{in} in the production of $M_{i(n+1)}$.

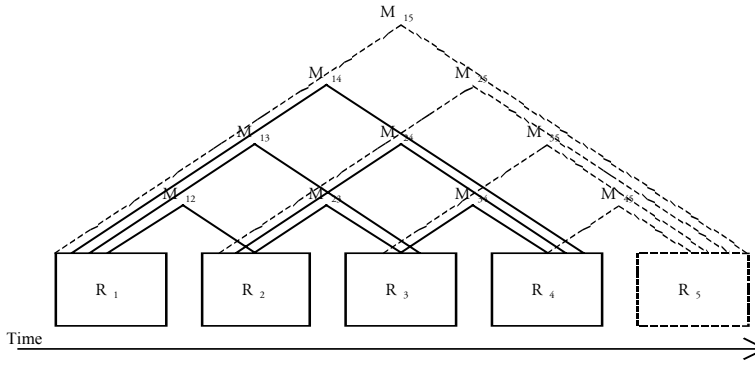


Figure 2: Rule and meta-rule sets

The observation we made in scenario B above presumes that previous meta-rule sets are available. This may not always be the case and depends on the mining strategy employed by the meta-miner. A possible preference might be that meta-rules are mined between an *elected* starting rule set (say R_1 of Figure 2) and the current one. Thus, one new meta-rule set is generated every time the current rule set is updated. In this case, scenario B still applies, as the meta-rule set between the starting rule set and the preceding one is available (for example, if R_5 is the new rule set and we are extracting M_{15}, the meta-rule set M_{14} is available). Problems can arise when in addition to this simplified meta-rule mining technique we also employ a 'time windowing' approach.

The time windowing approach to incremental rule updates was initially proposed by Rainsford, *et al.* (Rainsford, Mohania and Roddick 1997). Its purpose is to discard some old data during the generation of the current rule set, ensuring that its contents more closely represent the presently prevailing regularities in the environment. This addition to the incremental updating process can be viewed as an improvement on earlier work such as Cheung, *et al.* (1996), who highlight the need for statistically significant amounts of data. The time windowing approach retains this important aspect but maintains that some of the old data should no longer contribute to the current rules. We wish to follow a similar approach in producing meta-rule sets. That is, we would no longer keep meta-rules that span over a very long period of time.

In addition to applying time windowing, we may also request that to justify a meta-rule update we must have an appropriately significant change between the newly generated rule set and the previous one. For example, if a meta-rule set M_{ij} exists and there is very little change observed between rule sets R_j and its current

successor R_{j+1}, then there might be very little advantage gained by extracting $M_{i(j+1)}$. User defined thresholds may be used to control this process. For instance, a delta-file can be kept to store rules that have changed since the last meta-rule update, and a threshold (e.g. the percentage of changed rules compared to the total number of rules must exceed a given value) has to be reached to trigger the meta-rule update.

4 Discussions on a Framework for Mining Evolution Patterns in Spatio-Temporal Data

In earlier discussions, we gave a definition for a rule in knowledge discovery terms. The description *If A then B with caveat c* can be easily converted into more appropriate formats used in conjunction with popular data mining techniques, such as generalisation and association. An even simpler description is to express a general rule as a function of the data itself, e.g. *R(X)* where X represents the domain or the data set being mined and *R* is a data mining function. By extrapolating this train of thought, a meta-rule would then be a function of this function, e.g. *M(R(X))*. Using Clausal Form Logic and omitting the uncertainty measure *c* for the moment, a rule can be written as

$$A \Rightarrow B$$

or more specifically,

$$\{p_1, ..., p_m\} \Rightarrow \{q_1, ..., q_n\} \quad (1)$$

where $p_1, ..., p_m$ and $q_1, ..., q_n$ ($n, m \geq 0$) are atomic logical expressions, ie. either true or false values, or predicates with at least an argument each. In a spatial (and hence also spatio-temporal) environment, we can distinguish between three classes of predicates that describe attributes of spatial objects, their location and spatial relationships:

attribute predicates *a_predicate(o, v)*
positional predicates *p_predicate(o, v)*
relationship predicates *r_predicate(o, o')*

where *o* and *o'* denote spatial objects and *v* attribute values or positional references. We treat the first two types in the same way since for our purposes positional information can be managed as another set of attributes. A fourth type of predicate that is introduced by the temporal dimension concerns the existence of objects over time. At any given time,

existential predicates *e_predicate(o)*

describe the status of object *o* with regards to its presence in the corresponding temporal snapshot of the data.

Suppose that there exists a set of rules about the spatio-temporal environment at time T_1. As data collection continues, at time T_2 another set of rules may be extracted. A simple rule of the form:

$$p \Rightarrow q$$

where *p* and *q* are predicates of the above four classes may change in several different

ways. In Table 3 existential, value related and relationship predicates are separated because changes in values and object identities as well as the predicate itself indicate different forms of evolution in a rule. Notice that it is possible to use the same four groups to categorise rule sets at times T_1 and T_2 as has been done while discussing meta-rules. Rows with no explanations in Table 3 contain *new* rules with their *expired* counterparts in the earlier rule set, while the other explanations indicate *changed* rules unless it is explicitly stated that they are *unchanged*.

Predicate	Object	Value	Object	Explanation
U	U	-	-	Unchanged (existential) rule
U	C	-	-	-
U	U	U	-	Unchanged (value) rule
U	U	C	-	Value evolution
U	C	U	-	Object substitution
U	C	C	-	
U	U	-	U	Unchanged (relationship) rule
U	U	-	C	Object substitution
U	C	-	U	Object substitution
U	C	-	C	
C	U	-	-	Existential change
C	C	-	-	
C	U	-	U	Relationship change
C	U	-	C	
C	C	-	U	
C	C	-	C	

Table 3: Changes in rule description over time

It is also possible to observe other types of change in Equation 1. Instead of a variation in the descriptions, the structure of the antecedent or consequent may evolve. Suppose, for instance, that another term is added to the antecedent at time T_2 while the rest of the formula remains intact. Equation 1 can then be written as

$$\{p_1, ..., p_m, p_{(m+1)}\} \Rightarrow \{q_1, ..., q_n\} \quad (2)$$

This equates into a more specific description of the rule, or *specialisation*. For instance, an additional observation can be made regarding the object in the rule. When this is incorporated into the rule, our knowledge about the object is increased and hence becomes more specific. Similarly, if a term is removed from either the antecedent or the consequent, the expression becomes more general, ie. the evolution we observe is a form of *generalisation*.

Another aspect of evolution within a predicate term is the *rate of change* in the attribute value of an object. The multi-valued predicates that can be used are often qualitative (*e.g.* colour can be blue, white, direction can be north, north-east, etc.) and are not immediately quantifiable. It is nevertheless important to note just how much an attribute has changed from one time to another, *ie.* to define the proximity between two predicate values. A range of different norms can be employed to define distances

between values, both for numeric attributes and qualitative descriptions. An example for the qualitative attribute colour is to convert colour names to numbers on the hue palette using an auxiliary reference look-up table that uses a 360 degree disk representation for all allowed colours and take the difference of the angles corresponding to two colours.

In spatial and spatio-temporal information systems quantitative (and sometimes derived qualitative) data often contain measurement errors making subsequent observations for the same set of objects vary. Thus, rules that may indicate minor change over time should be treated with care if change does not exceed some given thresholds. In addition, change in a particular attribute may not be considered interesting. For example, a drop in the average temperature from summer to winter may be sizeable but not necessarily interesting. The determination of the attributes where change carries a high level of significance hence remains an important aspect of spatio-temporal evolution mining. We propose that the specification of these attributes be, at least partially, the a user responsibility since their relative importance is commonly application dependent. Moreover, the attributes relevant to interesting discoveries may change over time, and therefore need to be maintained between data mining processes. This observation is underlined by a discussion in (Silberschatz and Tuzhilin 1996) which highlights the relevance of objective and subjective interestingness measures and their tendency to evolve over time.

Furthermore, when an object parameter changes in a predicate expression, it may not necessarily imply a drastic deviation from the original rule. In a spatio-temporal environment, objects are not just related by spatial relationships and their positional characteristics, but depending on the model employed they often belong in a class structure of types. This means that when an object is replaced in a predicate by one of its descendants in a particular class structure, the rule becomes more *refined*, because more information is available about the object type. Conversely, if an object is replaced by one of its ancestors, the description becomes *coarser*. A paradox of these changes is that the particularity of an object and its support are inversely related, *ie.* the more refined a description the less likely that the corresponding object occurs in the database and hence some of the strength of the associated rules may diminish. On the other hand, other types of uncertainty measures, such as confidence values, may be unaffected or even improved.

In summary, a spatio-temporal rule extracted by data mining techniques can change in time, space, description and structure. If a rule contains an antecedent and a consequent, there are in total 2^8 possible combinations, some of which are not valid. For example, it is safe to require that both sides of the rule have to change in time in order to constitute any kind of change in the rule. On the other hand, if a rule does not change in time, then no other component should be allowed to change either. This "*no alternate realities*" requirement does not, however, exclude that two rules in a particular rule set cannot be very similar. For instance, it is possible that rule A of the set is a refined version of another rule B of the set, but as far as their membership

in the rule set goes, they are considered different. For the remaining combinations, most constitute change to a degree where rules would be regarded as new and not altered, leaving only a small number of options interesting from the knowledge discovery point of view (although the fact that a rule has not changed but was replaced entirely could be an interesting observation in itself).

This paper outlines the work in progress of a part of a research project investigating the applicability of data mining in a variety of domains. It is our contention that many of the useful aspects of data mining are in the mining of the rules themselves and we argue that meta-rule mining can yield useful results in this area.

Further work is progressing in the areas of temporal, spatio-temporal, incremental and meta-rule mining and results to date can be found at http://www.cis.unisa.edu.au.

5 References

Abraham, T. and Roddick, J.F. 1997a. 'Discovering Meta-rules in Mining Temporal and Spatio-Temporal Data'. In *Proc. Eighth International Database Workshop*, 30-41.

Abraham, T. and Roddick, J.F. 1997b. 'Research issues in spatio-temporal knowledge discovery'. In *Proc. SIGMOD'97 Workshop on Data Mining*, 85.

Abraham, T. and Roddick, J.F. 1998a. 'Incremental meta-mining from large temporal data sets: extended report'. Technical Report University of South Australia.

Abraham, T. and Roddick, J.F. 1998b. 'Opportunities for knowledge discovery in spatio-temporal information systems'. *Australian Journal of Information Systems.* **5**(2):3-12.

Cheung, D.W., Han, J., Ng, V.T. and Wong, C.Y. 1996. 'Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique.'. In *Proc. International Conference on Data Engineering (ICDE'96)*, New Orleans, Louisiana, USA.

Cheung, D.W., Lee, S.D. and Kao, B. 1997. 'A General Incremental Technique for Maintaining Discovered Association Rules'. In *Proc. Fifth International Conference on Database Systems for Advanced Applications*, Melbourne, Australia.

Cheung, D.W., Ng, V.T. and Tam, B.W. 1996. 'Maintenance of Discovered Knowledge : A Case in Multi-level Association Rules'. In *Proc. Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon. AAAI Press, Menlo Park, California. 307-310.

Han, J., Cai, Y. and Cercone, N. 1992. 'Knowledge Discovery in Databases: An Attribute-Oriented Approach'. In *Proc. Eighteenth International Conference on Very Large Data Bases*, Vancouver, Canada.

Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H. and Verkamo, A.I. 1994. 'Finding interesting Rules from Large Sets of Discovered Association Rules'. In *Proc. Third International Conference on Information and Knowledge Management*, Gaithersburg, Maryland. ACM Press. 401-407.

Koperski, K. and Han, J. 1995. 'Discovery of Spatial Association Rules in Geographic Information Databases'. In *Proc. Fourth International Symposium on Large Spatial Databases*, Maine. 47-66.

Lee, S.D. and Cheung, D.W. 1997. 'Maintenance of Discovered Association Rules: When to update?'. In *Proc. 1997 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97)*, Tucson, Arizona. 51-58.

Lu, W., Han, J. and Ooi, B.C. 1993. 'Discovery of General Knowledge in Large Spatial Databases'. In *Proc. 1993 Far East Workshop on GIS (IEGIS 93)*, Singapore. 275-289.

Rainsford, C.P., Mohania, M.K. and Roddick, J.F. 1996. 'Incremental maintenance techniques

for discovered classification rules'. In *Proc. International Symposium on Cooperative Database Systems for Advanced Applications*, Kyoto, Japan. World Scientific Publishing Co. 302-305.

Rainsford, C.P., Mohania, M.K. and Roddick, J.F. 1997. 'A temporal windowing approach to the incremental maintenance of association rules'. In *Proc. Eighth International Database Workshop (IDW'97)*, 78-94.

Rainsford, C.P. and Roddick, J.F. 1996. 'Temporal data mining in information systems: a model'. In *Proc. Seventh Australasian Conference on Information Systems*, Hobart, Tasmania. **2**:545-553.

Silberschatz, A. and Tuzhilin, A. 1996. 'What Makes Patterns Interesting in Knowledge Discovery Systems'. *IEEE Trans. Knowl. and Data Eng.* **8**(6):970-974.

On the Suitability of Genetic-Based Algorithms for Data Mining*

Sunil Choenni[1,2]

[1] Nat. Aerospace Lab., NLR, P.O. Box 90502, 1006 BM Amsterdam, The Netherlands
[2] Univ. of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
choenni@nlr.nl

Abstract. Data mining has as goal to extract knowledge from large databases. A database may be considered as a search space consisting of an enormous number of elements, and a mining algorithm as a search strategy. In general, an exhaustive search of the space is infeasible. Therefore, efficient search strategies are of vital importance. Search strategies on genetic-based algorithms have been applied successfully in a wide range of applications. We focus on the suitability of genetic-based algorithms for data mining. We discuss the design and implementation of a genetic-based algorithm for data mining and illustrate its potentials.

1 Introduction

Research and development in data mining evolves in several directions, such as association rules, time series, and classification. The latter field has our attention. We have developed an algorithm to classify tuples in groups and to derive rules from these groups. In our view, a user formulates a mining question and the algorithm selects the group(s) that satisfy this question. For example, in an insurance environment, a question may be to identify persons with (more than average) chances of causing an accident. Then, the algorithm searches for the (group) profiles of these persons.

In general, the search spaces that should be inspected in order to find answers on mining questions are very large, making exhaustive search infeasible. So, heuristic search strategies are of vital importance to data mining. Genetic algorithms, which are heuristic search strategies, have been successfully used in a wide range of applications. A genetic algorithm is capable of exploring different parts of a search space [10].

In this paper, we discuss the applicability of a genetic-based algorithm to the search process in data mining. We show how a genetic algorithm can be suited for data mining problems. In our approach, a search space consists of expressions. An expression is a conjunction of predicates and each predicate is defined on a database attribute. Initially, a random number of expressions, called initial population, is selected. Then, the initial population is manipulated

* This research has been sponsored by the Dutch Ministry of Defense.

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 55–68, 1999.

by applying a number of operations. The best individuals are selected to form the next generation and the manipulation process is repeated until no significant improvement of the population can be observed.

In general, data mining algorithms require a technique that partitions the domain values of an attribute in a limited set of ranges, simply because considering all possible ranges of domain values is infeasible. Suppose that we have an attribute *age* which has a domain between 18 to 65, and an expression of the form *age* **in** $[v_i, v_k]$, in which v_i and v_k are values from the domain of *age*, defining a range of values. The problem is how to choose the values for v_i and v_k. As illustrated in [11], this is in general an NP-complete problem. Our solution to this problem is based on a suitable choice of the mutation operator (see Section 3.3). Furthermore, we have chosen a representation for individuals that seamlessly fits in the field of databases. The same holds for the manipulation operators and the function to rank individuals (fitness function). The fitness function discussed in this paper is close to our intuition and gives rise to a speed up of the optimization process. Based on our approach, we have implemented a (prototype) tool for data mining, and have performed a preliminary evaluation. The results will be presented in this paper.

A genetic approach has been proposed in [2] to learn first order logic rules and in [7] a framework is proposed for data mining based on genetic programming. However, the authors neither come up with a implementation nor with experiments. The effort in [2] is focussed towards machine learning, and the important data mining issue of integration with databases is superficially discussed. The effort in [7] describes a framework for data mining based on genetic programming, and stresses on the integration of genetic programming and databases. However, an elaborated approach to implement and evaluate the framework is not presented. Other related research has been reported in [1,8,9]. While in [8,9] variants of a hill climber are used to identify the group(s) of tuples satisfying a mining question, the approach in [1] is based on decision trees. However, the problem of partitioning attribute values has not been discussed in these efforts. We note that a genetic-based algorithm has, by nature, a better chance to escape from a local optimum than a hill climber.

The remainder of this paper is organized as follows. In Section 2, we outline some preliminaries and problem limitations. In Section 3, we identify the issues that play a role in genetic-based algorithms and adapt them in a data mining context. In Section 4, we point out a number of rules that may speed up the search process of a genetic-based algorithm. Section 5 is devoted to an overall algorithm for data mining. In Section 6, we discuss the implementation of the algorithm and some preliminary results. Finally, Section 7 contains conclusions and further work.

2 Preliminaries & Problem Limitations

In the following, a database consists of a universal relation [6]. The relation is defined over some independent single valued attributes, such as att_1,att_2,..., att_n,

and is a subset of the Cartesian product $\mathrm{dom}(att_1) \times \mathrm{dom}(att_2) \times ... \times \mathrm{dom}(att_n)$, in which $\mathrm{dom}(att_j)$ is the set of values that can be assumed by attribute att_j. A tuple is an ordered list of attribute values to which a unique identifier (tid) is associated. So, we do not allow missing attribute values. Furthermore, we assume that the content of the database remains the same during the mining process.

An expression is used to derive a relation, and is defined as a conjunction of predicates over some attributes. The length of an expression is the number of attributes involved in the expression. An example of an expression of length 2 is (*age* **in** $[19, 24] \wedge$ *gender* **is** 'male'), representing the males who are older than 18 and younger than 25. An expression with length 1 is called an *elementary* expression. In this paper, we deal with search spaces that contain expressions.

3 Data Mining with Genetic Algorithms

Initially, a genetic algorithm [10] randomly generates an initial population. Traditionally, individuals in the population are represented as bit strings. The quality of each individual, i.e., its fitness, is computed. On the basis of these qualities, a selection of individuals is made (an individual may be chosen more than once). Some of the selected individuals undergo a minor modification, called mutation. For some pairs of selected individuals a random point is selected, and the substrings behind this random point are exchanged; this process is called cross-over. The selected individuals, modified or not, form a new population and the same procedure is applied to this generation until some predefined criteria are met.

In the following, we discuss the issues that play a role in tailoring a genetic algorithm for data mining. Section 3.1 is devoted to the representation of individuals and Section 3.2 to the fitness function. Finally, in Section 3.3, we discuss the two operators to manipulate an individual.

3.1 Representation

An individual is regarded as an expression to which some restrictions are imposed with regard to the the notation of elementary expressions and the number of times that an attribute may be involved in the expression. The notation of an elementary expression depends on the domain type of the involved attribute. If there exists no ordering relationship between the attribute values of an attribute *att*, we represent an elementary expression as follows: *expression* := *att* **is** (v_1, v_2,...,v_n), in which $v_i \in \mathrm{dom}(att)$, $1 \leq i \leq n$. In this way, we express that an attribute *att* assumes one of the values in the set $\{v_1, v_2, ..., v_n\}$. If an ordering relationship exists between the domain values of an attribute, an elementary expression is denoted as *expression* := *att* **in** $[v_i, v_k]$, $i \leq k$, in which $[v_i, v_k]$ represents the values within the range of v_i and v_k. An attribute is allowed to participate at most once in an individual. This restriction is imposed to prevent the exploration of expressions to which no tuples satisfy. In the following, an expression to which no tuples qualify will be called an *empty* expression. Consider a database in which, among others, the age of persons is recorded. Then, the expression *age* **in** [19,34] $\wedge$ *age* **in** [39,44] represents the class of persons whose

p_1 = *gender* **is** ('male') $\wedge$ *age* **in** [19,34]
p_2 = *age* **in** [29,44] $\wedge$ *town* **is** ('Almere', 'Amsterdam', 'Weesp') $\wedge$ *category* **is** ('lease')
p_3 = *gender* **is** ('male') $\wedge$ *age* **in** [29,34] $\wedge$ *category* **is** ('lease')
p_4 = *gender* **is** ('female') $\wedge$ *age* **in** [29,40] $\wedge$ *category* **is** ('lease') $\wedge$ *price* **in** [50K,100K]
p_5 = *gender* **is** ('male') $\wedge$ *price* **in** [20K,45K]

Fig. 1. Example of a population

age is between 19 and 34 as well as between 39 and 44. It should be clear that no persons will satisfy this expression, since *age* is a single-valued attribute.

In the following, a population is defined as a set of individuals. As a running example, we use a database that keeps record of cars and their owners. This artificial database consists of the following universal relation[1]: *Ex(gender, age, town, category, price, damage)*, in which the attributes *gender*, *age*, and *town* refer to the owner and the remainder of the attributes refer to the car. Attribute *category* records whether a car is leased or not, and *damage* records whether a car has been involved in an accident or not. An example of a population consisting of 5 individuals is given in Figure 1.

3.2 Fitness Function

A central instrument in a genetic algorithm is the fitness function. Since a genetic algorithm is aimed to the optimization of such a function, this function is one of the keys to success. Consequently, a fitness function should represent all issues that play a role in the optimization of a specific problem. Before enumerating these issues in the context of data mining, we introduce the notion of cover.

Definition 1: Let D be a database and p an individual defined on D. Then, the number of tuples that satisfies the expression corresponding to p is called the cover of p, and is denoted as $\|\sigma_p(D)\|$. The set of tuples satisfying p is denoted as $\sigma_p(D)$.

Note that p can be regarded as a description of a class in D and $\sigma_p(D)$ summarizes the tuples satisfying p. Within a class we can define subclasses. In the following, we regard classification problem as follows: *Given a target class t, search interesting subclasses, i.e., individuals, within class t*. We note that the target class is the class of tuples in which interesting knowledge should be searched for. Suppose we want to expose the profiles of risky drivers, i.e., the class of persons with (more than average) chances of causing an accident, from the database *Ex(gender, age, town, category, price, damage)*. Then, these profiles should be searched for in a class that records the characteristics of drivers that caused accidents. Such a class may be described as *damage* = 'yes'.

We feel that the following issues play a role in classification problems.

- The cover of the target class. Since results from data mining are used for informed decision making, knowledge extracted from databases should be

[1] We note that a universal relation can be obtained by performing a number of joins between the relations involved in a database.

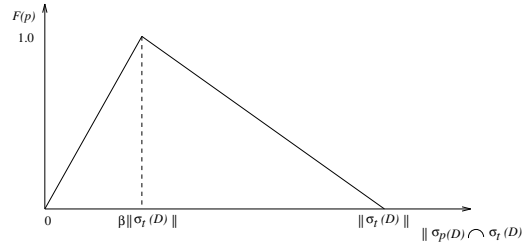


Fig. 2. Shape of the fitness function

supported by a significant part of the database. This increases the reliability of the results. So, a fitness function should take into account that small covers are undesired.

– The ratio of the cover of an individual p to the cover of the target class t, i.e., $\frac{\|\sigma_p(D)\cap\sigma_t(D)\|}{\|\sigma_t(D)\|}$. If the ratio is close to 0, this means that only a few tuples of the target class satisfy individual p. This is undesired for the same reason as a small cover for a target class. If the ratio is close to 1, almost all tuples of the target class satisfy p. This is also undesired because this will result in knowledge that is often known. A fitness function should take these properties into account.

Taking into account above-mentioned issues, we have defined the following fitness function:

$$F(p) = \begin{cases} \frac{\|\sigma_p(D)\cap\sigma_t(D)\|}{\beta\|\sigma_t(D)\|}C(t) & \text{if } \|\sigma_p(D)\cap\sigma_t(D)\| \leq \beta\|\sigma_t(D)\| \\ \\ \frac{\|\sigma_p(D)\cap\sigma_t(D)\|-\|\sigma_t(D)\|}{\|\sigma_t(D))\|(\beta-1)}C(t) & \text{otherwise} \end{cases}$$

in which $0 < \beta \leq 1$, and

$$C(t) = \begin{cases} 0 \text{ if } \frac{\|\sigma_t(D)\|}{\|\sigma(D)\|} \leq \alpha \\ 1 \text{ otherwise} \end{cases}$$

and $0 \leq \alpha \leq 1$.

We note that the values for α and β should be defined by the user and will vary for different applications. The value of α defines the fraction of tuples that a target class should contain in order to be a candidate for further exploration. The value β defines the fraction of tuples that an individual should represent within a target class in order to obtain the maximal fitness. In Figure 2, the shape of the fitness function is presented.

The fitness grows linearly with the number of tuples satisfying the description of an individual p as well as satisfying a target class t, i.e., $\|\sigma_p(D) \cap \sigma_t(D)\|$, above a user-defined value α, and decreases linearly with $\|\sigma_p(D) \cap \sigma_t(D)\|$ after reaching the value $\beta\|\sigma_t(D)\|$.

It should be clear that our goal is to search for those individuals that approximate a fitness of $\beta\|\sigma_t(D)\|$. Consider the target class *damage = yes* that consists

of 100.000 tuples. Assume that a profile is considered risky if about 30.000 out of 100.000 persons satisfy this profile. This means that $\beta \approx 0.3$. Assuming that 33.000 of the persons caused an accident are young males, the algorithm should find individuals like (*gender* **is** ('male') $\wedge$ *age* **in** [19,28]).

3.3 Manipulation Operators

Mutation As stated in the introduction of this section, a mutation modifies an individual. In defining the mutation operator, we take into account the domain type of an attribute. If there exists no ordering relationship between the domain values, then we select randomly an attribute value and replace it by another value, which can be a NULL value as well, in an expression that contains this attribute. For example, a mutation on attribute town of individual p_2 (see Figure 1) may result into $p_2' =$ *age* **in** [29,44] $\wedge$ *town* **is** ('Almere', 'Den Haag', 'Weesp') $\wedge$ *category* **is** ('lease').

If there exists a relationship between the domain values of an attribute, the mutation operator acts as follows in the case that a single value is associated with this attribute in an expression, i.e., the expression looks as *att* **is** (v_c). Let $[v_b, v_e]$ be the domain of attribute *att*. In order to mutate v_c, we choose randomly a value $\delta_v \in [0, (v_e - v_b)\mu]$, in which $0 \leq \mu \leq 1$. The mutated value v_c' is defined as $v_c' = v_c + \delta_v$ or $v_c' = v_c - \delta_v$ as long as $v_c' \in [v_b, v_e]$. The parameter μ is used to control the maximal increase or decrease of an attribute value.

To handle overflow, i.e., if $v_c' \notin [v_b, v_e]$, we assume that the successor of v_e is v_b, and, consequently the predecessor of v_b is v_e. To compute a mutated value v_c' appropriately, we distinguish between whether v_c will be increased or decreased, which is randomly determined.

In the case that v_c is increased

$$v_c' = \begin{cases} v_c + \delta_v & \text{if } v_c + \delta_v \in [v_b, v_e] \\ v_b + \delta_v - (v_e - v_c) & \text{otherwise} \end{cases}$$

and in the case v_c is decreased

$$v_c' = \begin{cases} v_c - \delta_v & \text{if } v_c - \delta_v \in [v_b, v_e] \\ v_e - \delta_v + (v_c - v_b) & \text{otherwise} \end{cases}$$

Let us consider the situation in which more than one value is associated with an attribute *att* in an expression. If a list of non successive (enumerable) values is associated with *att*, we select one of the values and compute the new value according to one of the above-mentioned formulas. If a range of successive values, i.e., an interval, is associated with *att*, we select either the lower or upper bound value and mutate it. A potential disadvantage of this strategy for intervals is that an interval may be significantly enlarged, if the mutated value crosses a domain boundary. Suppose that the domain of *age* is [18,60], and we mutate the upper bound value of the expression *age* **in** $[55, 59]$, i.e., the value 59. Assume that the value 59 is increased by 6, then 59 is mutated in the value 23. The new expression becomes *age* **in** $[23, 55]$.

We note that the partitioning of attribute values, i.e., the selection of proper intervals in an expression, is simply adjusted by the mutation operator.

Cross-over The idea behind a crossover operation is as follows; it takes as input 2 expressions, selects a random point, and exchanges the subexpressions behind this point. In general, not all attributes will be involved in an expression. This may have some undesired effects for a cross-over. First, a cross-over may produce individuals in which an attribute is involved more than once. For example, a cross-over between the individuals $p_1 =$ *gender* **is** ('male') $\wedge$ *age* **in** [19,34] and $p_2 =$ *age* **in** [29,44] $\wedge$ *town* **is** ('Almere', 'Amsterdam', 'Weesp') $\wedge$ *category* **is** ('lease') after the first attribute results into the following individuals: $p'_1 =$ *gender* **is** ('male') $\wedge$ *town* **is** ('Almere', 'Amsterdam', 'Weesp') $\wedge$ *category* **is** ('lease') and $p'_2 =$ *age* **in** [19,34] $\wedge$ *age* **in** [29,44]. As we can see, the attribute *age* appears twice in p'_2.

Second, a cross-over may result in an offspring that is exactly the same as the parents with probability 1.0. For example, a cross-over between p_1 and p_2 that occurs on a point that is beyond the last elementary expression of p_2, i.e., *category* **is** ('lease'), will result into the equal new individuals.

To prevent the above-mentioned effects, we apply the following technique to perform cross-overs. Consider two individuals p_i and p_j that have been selected for crossover. Let A_i be the set of attributes that is not involved in p_i but is involved in p_j and A_j the set of attributes that is not involved in p_j but is involved in p_i. Then, for each attribute in A_i, we generate an empty elementary expression using this attribute and add it to p_i. The same procedure is applied to the attributes of A_j. This procedure has as effect that the lengths of p_i and p_j become equal. Finally, we regard an individual as a sequence of elementary expressions, and order these in p_i and p_j according to the rule that elementary expressions having the same attribute will appear at the same position in p_i and p_j. Then, the cross-over between p_i and p_j can be performed. We note that the cross-over point should be chosen between the first and final position of p_i or p_j. The following example illustrates this technique.

Example 1 Consider the individuals $p_1 =$ *gender* **is** ('male') $\wedge$ *age* **in** [19,34] and $p_2 =$ *age* **in** [29,44] $\wedge$ *town* **is** ('Almere', 'Amsterdam', 'Weesp') $\wedge$ *category* **is** ('lease') again. Then, $A_1 = \{$*town*, *category*$\}$ and $A_2 = \{$*gender*$\}$. So, we extend p_1 with the following expression *town* **is** ('') $\wedge$ *category* **is** ('') and p_2 is extended with *gender* **is** ('').

After ordering the elementary expressions p_1 and p_2 look as follows:
$p_1 =$ *gender* **is** ('male') $\wedge$ *age* **in** [19,34] $\wedge$ *town* **is** ('') $\wedge$ *category* **is** ('')
$p_2 =$ *gender* **is** ('') $\wedge$ *age* **in** [29,44] $\wedge$ *town* **is** ('Almere', 'Amsterdam', 'Weesp') $\wedge$ *category* **is** ('lease')
Now, a cross-over at position 2 results into
$p'_1 =$ *gender* **is** ('male') $\wedge$ *age* **in** [19,34] $\wedge$ *town* **is** ('Almere', 'Amsterdam', 'Weesp') $\wedge$ *category* **is** ('lease')
$p'_2 =$ *gender* **is** ('') $\wedge$ *age* **in** [29,44] $\wedge$ *town* **is** ('') $\wedge$ *category* **is** ('')
Note that p'_2 is equal to *age* **in** [29,44]. □

In the next section, we introduce a number of rules that may prevent the exploration of unpromising individuals.

4 Optimization Rules

In this section, we discuss two propositions that may be used to prevent the exploration of unprofitable individuals. These propositions are derived from the shape of the fitness function. The complexity of a genetic-based algorithm for data mining is determined by the evaluation of the fitness function [5], since this is computationally the most expensive operation. Before presenting these propositions, we introduce the notion of a similar of an individual.

Definition 2: Let length(p) be the number of elementary expressions involved in p. An individual p_{sim} is a *similar* of p if each elementary expression of p_{sim} is contained in p or p_{sim} contains each elementary expression of p and length(p_{sim}) $\neq$ length(p).

As stated in the foregoing, we search for individuals with high values for the fitness function F, see section 3.2. for F.

We note that the computation of $F(p)$ requires the number of tuples that satisfy individual p. So, these tuples should be searched for and retrieved from the database, which is a costly operation [6]. Although several techniques may be used to minimize the number of retrievals from a database, still large amounts of tuples have to be retrieved from the database in mining applications. The techniques to minimize retrievals are mainly based on storing frequently used tuples in an efficient way in main memory [4]. In this way, disk accesses are reduced.

In the following, two propositions will be presented that may be used to avoid the computation of fitness values of unprofitable individuals. These propositions decide if the fitness value of a similar of an individual p is worse than the fitness of p. If this is the case, this similar can be excluded from the search process.

Proposition 1: Let p_{sim} be a similar of p. If $\|\sigma_p(D) \cap \sigma_t(D)\| \leq \beta\|\sigma_t(D)\|$ and length(p_{sim}) $>$ length(p) then $F(p_{sim}) \leq F(p)$.

Proof. From length(p_{sim}) $>$ length(p) follows that $\sigma_{p_{sim}}(D) \subseteq \sigma_p(D)$. As a consequence, $\|\sigma_{p_{sim}}(D) \cap \sigma_t(D)\| \leq \|\sigma_p(D) \cap \sigma_t(D)\|$. Since $\|\sigma_p(D) \cap \sigma_t(D)\| \leq \beta\|\sigma_t(D)\|$, it follows $F(p_{sim}) \leq F(p)$. □

Proposition 2: Let p_{sim} be a similar of p. If $\|\sigma_p(D) \cap \sigma_t(D)\| \geq \beta\|\sigma_t(D)\|$ and length(p_{sim}) $<$ length(p) then $F(p_{sim}) \leq F(p)$.

Proof. Similar to the proof of Proposition 1. □

Note that the propositions do not require additional retrievals from a database to decide if $F(p_{sim}) \leq F(p)$.

We discuss an alternative how these propositions at cross-over level may contribute in optimizing the search process. As stated in the foregoing, a cross-over is applied on a mating pair and results into two offsprings. Suppose that a

mutation is performed after a cross-over, and the parent and the offspring with the highest fitness values are eligible to be mutated (see Section 5). Consider an offspring p_o resulted from a cross-over, and let p_o be a similar of p, one of its parents. If we can decide that $F(p_o) \leq F(p)$, then it is efficient to mutate p_o. The reason is that computation on an unmutated p_o will be a wasting of effort.

In the next section, we propose an overall algorithm, in which we apply the two propositions.

5 Algorithm

The previous section was devoted to the major issues that play a role in designing a genetic-based algorithm for data mining. In this section, we describe the overall algorithm. Before starting this description, we discuss a mechanism to select an individual for a next generation.

The mechanism to select individuals for a new generation is based on the technique of elitist recombination [12]. According to this technique, the individuals in a population are randomly shuffled. Then, the cross-over operation is applied on each mating pair, resulting into two offsprings. The parent and the offspring with the highest fitness value are selected for the next generation. In this way, there is a direct competition between the offsprings and their own parents. Note, the offspring provides the possibility to explore different parts of a search space.

The elitist recombination technique has been chosen for two reasons. First, there is no need to specify a particular cross-over probability, since each individual is involved in exactly one cross-over. Second, there is no need for intermediate populations in order to generate a new population as is the case in a traditional genetic algorithm. These properties simplify the implementation of a genetic algorithm. Let us outline the overall algorithm.

The algorithm starts with the initialization of a population consisting of an even number of individuals, called $P(t)$. The individuals in this population are shuffled. Then, the cross-over operation is applied on two successive individuals. After completion of a cross-over, the fitness values of the parents are compared[2]; the parent with the highest value is selected and it may be mutated with a probability c. This parent, p'_{sel}, is added to the next generation, and in case it is mutated its fitness value is computed. Then, for each offspring, p_o, we test if this offspring is a similar of p'_{sel} and if its fitness value is worse or equal than p'_{sel}. If this is true, p_o is an unpromising individual, and, therefore, we always mutate p_o. Otherwise, we mutate p_o with probability c. Note, to compare the fitness value between p'_{sel} and p_o, the propositions of the previous section are used. So, no additional fitness values are computed for this comparison. After possible mutation of the offsprings, their fitness values are computed, and the most fittest offspring is added to the new generation. This process is repeated for all individuals in a generation.

[2] These values are already computed and stored by the algorithm.

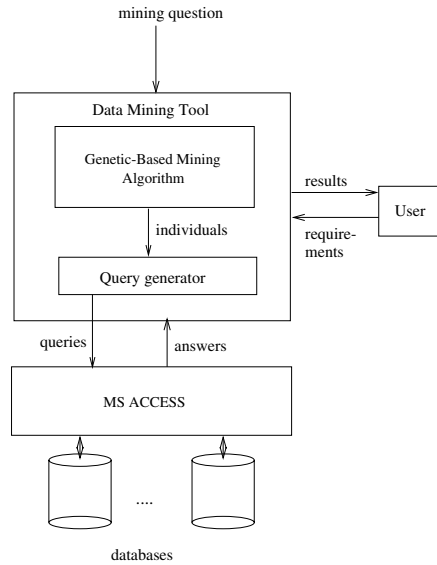


Fig. 3. Architecture of a data mining tool

Once the new population has been built up, the total fitness of the existing as well as of the new population is computed, and compared. The algorithm terminates if the total fitness of the new population does not significantly improve compared with the total fitness of the existing population, i.e., that the improvement of the total fitness of the new population is less than a threshold value ϵ. For a detailed discussion with regard to the algorithm, we refer to [5].

6 Implementation and Preliminary Results

Based on the algorithm described in Section 5, we have built a re-targetable prototype of a data mining tool, which means that the tool can be coupled to different databases. The main goal of the current effort is to determine if the genetic-based algorithm is able to find hidden knowledge.

Let us continue with the description of the tool. The tool takes as input a mining question and produces a set of answers for the question. The prototype is running in a Microsoft Access 97 environment that uses the Microsoft Jet Engine[3]. The genetic-based algorithm is implemented in Visual Basic. We have chosen this environment for two reasons. First, this environment is available at our laboratory. Second, the database that we want to mine is a Microsoft database.

In Figure 3, the architecture of our tool is represented. Once the tool receives a mining question, it runs the genetic-based mining algorithm, which generates, among others, a population. The individuals of the population are passed to the

[3] Within the scope of a feasibility study, a previous version of the algorithm was implemented in C and connected to the Monet Database Server [3].

Query Generator. For each individual a corresponding SQL query is generated. These queries are passed to the MS ACCESS DBMS, which on its turn processes the queries and passes the results to the data mining tool. These results are used to compute the fitness of each individual. Upon request of the user individuals and their associated fitness values can be shown. The user has the possibility to modify the initial mining question or to specify additional requirements with regard to the question. We note that this is a very useful feature of a mining tool, since, in practice, a user starts a mining session with a rough idea of what information might be interesting, and during the mining session (with help of the results provided by the system) the user specifies more precisely what information should be searched for.

The generation of a query corresponding to an individual p is straightforward. Recall that an individual is a conjunction of predicates. So, this forms the major part of the WHERE clause of a query corresponding to p. Since we require the number of tuples satisfying p and a target class t, the query corresponding to p is: *select count(*) from* database *where* $p \wedge t$.

We have applied the tool on an artificial database, and are currently applying it on a real-life database. In the following, we describe the databases and the results obtained so far.

Artificial database This database consists of the relation *Ex(gender, age, town, category, price, damage)*. For this database 100.000 tuples have been generated, of which 50% have a value "yes" for *damage*, i.e., 50% of the tuples relate to an accident. Furthermore, the fact that young men in lease cars have more than average chances to cause an accident was hidden in the database. The goal of mining this database was to determine whether the tool is capable to find the hidden fact. Therefore, we have set the target class as *damage* = 'yes', and we searched for the profile of risky drivers. We note that the expression for the hidden profile is: *age* **in** [19,24] $\wedge$ *category* **is** ('lease') $\wedge$ *gender* **is** ('male').

We have mined the database with varying initial populations, consisting of 36 individuals. The following three classes of initial population were distinguished: (1) random: the populations contained a few individuals that could set the algorithm quickly on a promising route, (2) modified random: individuals that could apparently set the algorithm on a promising route were replaced by other (not promising) individuals, and (3) bad converged: the populations contained individuals with low fitness values.

We have observed that the algorithm usually finds near optimal solutions, i.e., profiles that look like the hidden one, in less than 1000 fitness evaluations. The differences between the hidden profile and profiles found by the algorithm (for different initial populations) were mainly caused by variations in the range of attribute *age*.

With regard to the settings of the parameters α and β, we note that appropriate values could be easily selected, since the content of the database is precisely known. □

Real-life database Currently, we are mining a real-life database, the so-called FAA incident database, which is available at our aerospace laboratory. This database contains aircraft incident data that are recorded from 1978 to 1995. Incidents are potentially hazardous events that do not meet the aircraft damage or personal injury thresholds as defined by American National Transportation Safety Board (NTSB). For example, the database contains reports of collisions between aircraft and birds while on approach to or departure from an airport. The FAA database consists of more than 70 attributes and about 80.000 tuples.

The initial mining task on the FAA database was: search for the class of flights with (more than average) chances of causing an incident, i.e., profiles of risky flights. This search resulted in (valid) profiles but which could be easily declared. An example of such a profile is that aircraft with 1 or 2 engines are more often involved in incidents. The explanation for this profile is that these types of aircraft perform more flights.

During the mining process the mining question was refined in the following three more specific questions: (1) given the fact that an incident was due to operational defects not inflicted by the pilot, what is the profile of this type of incident?, (2) given the fact that an incident was due to mistakes of the pilot, what is the profile of this type of incident?, and (3) given the fact that an incident was due to improper maintenance, what is the profile of this type of incident?

We have proposed these questions to our tool with the following values for the parameters, $\alpha = 0$, $\beta = 0.25$, mutation probability (c) $= 0.3$, and $\epsilon = 0.1$. Furthermore, the population size was set on 50. On the first glance, the results of the tool appear to be promising. Safety experts at our laboratory are analysing the results. On the basis of their analysis, we will set up a plan to mine the database more systematically, and to study the impact of different parameter values on the results provided by the tool. The goal of the latter study is to formulate some guidelines for selecting parameter values for similar type of databases, such as an aircraft accident database. □

Although our evaluation is not completed yet and a significant amount of research has to be done, e.g., on performance issues, in order to build an adequate genetic-based data mining tool, the preliminary results are promising. A second observation is that the range interval of an attribute in an expression may significantly enlarged, if a mutation occurs on a domain boundary. An interval that consists of (almost) the whole the domain slows down the search process. In a next version of the tool, we will enhance the mutation operator. An alternative is to clip on boundary values in cases of overflow.

7 Conclusions & Further Research

In order to answer mining questions, very large search spaces should be inspected, making an exhaustive search infeasible. So, heuristic search strategies are of vital importance in searching such spaces. We have discussed a genetic-based algorithm that may be used for data mining. Contrary to the conventional bit

string representation in genetic algorithms, we have chosen a representation that fits better in the field of databases. The fitness function discussed in this paper is close to our intuition and gives rise to an optimization of the search process.

A genetic-based algorithm for data mining has two major advantages. First, the problem of partitioning attribute values in proper ranges could be solved by choosing a suitable mutation operator. Second, a genetic-based algorithm is able to escape a local optimum and does not pose any restrictions on the structure of a search space.

By means of a (prototype) implementation and a preliminary evaluation, we have shown the potentials of a genetic-based data mining tool. Since the preliminary results of the tool appear to be promising, we are setting up a research plan to evaluate this tool thoroughly. The outcome of the evaluation will determine our future research activities in this field.

Acknowledgements The author is grateful to Wim Pelt from the Dutch Ministry of Defense, who made this research possible. Hein Veenhof and Egbert Boers from NLR are thanked for their valuable comments on earlier drafts of this paper. Finally, Leo de Penning and Martijn Suurd from the Univ. of Twente are thanked for their implementation efforts.

References

1. Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., Swami, A., An Interval Classifier for Database Mining Applications, Proc. 18th Int. Conf. Very Large Data Base, pp. 560-573. 56
2. Augier, S., Venturini, G., Kodratoff, Y., Learning First Order Logic Rules with a Genetic Algorithm, Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining, pp. 21-26. 56
3. Boncz, P., Wilschut, A., Kersten, M., Flattening an Object Algebra to Provide Performance, Proc. 14th Int. Conf. on Data Engineering, pp. 568-577. 64
4. Choenni, R., Siebes, A., Query Optimization to Support Data Mining, Proc. DEXA '97 8th Int. Workshop on Database and Expert Systems Applications, pp. 658-663. 62
5. Choenni, R., On the Suitability of Genetic-Based Algorithms for Data Mining, extended version, to appear as NLR technical publication. 62, 64
6. Elmasri, R., Navathe, S., Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company, 1989. 56, 62
7. Freitas, A., A Genetic Programming Framework for two Data Mining Tasks: Classification and Generalized Rule Induction, Proc. Int. Conf on Genetic Programming 1997, pp. 96-101. 56
8. Han, J., Cai, Y., Cerone, N., Knowledge Discovery in Databases: An Attribute-Oriented Approach, in Proc. 18th Int. Conf. Very Large Data Base, pp. 547-559. 56
9. Holsheimer, M., Kersten, M.L., Architectural Support for Data Mining, Proc. AAAI-94 Workshop on Knowledge Discovery, pp. 217-228. 56
10. Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York, USA. 55, 57

11. Srikant, R., Agrawal, R., Mining Quantitative Association Rules in Large Relational Tables, Proc. ACM SIGMOD'96 Int. Conf. Management of Data, pp. 1-12. 56
12. Thierens, D., Goldberg, D., Elitist Recombination: an integrated selection recombination GA, 1st IEEE Conf. on Evolutionary Computing, pp. 508-512. 63

Data Visualization in a Web Warehouse*

S. S. Bhowmick, S. K. Madria, W.-K. Ng, and E.-P. Lim

Center for Advanced Information Systems, School of Applied Science
Nanyang Technological University, Singapore 639798, Singapore
{sourav,askumar,wkn,aseplim}@cais.ntu.edu.sg

Abstract. The effective representation and manipulation of web data is currently an active area of research in databases. In a web warehouse, *web information coupling* provides the means to derive useful information from the WWW. Web information is materialized in the form of *web tuples* and stored in *web tables*. In this paper, we discuss web data visualization operators such as *web nest*, *web coalesce*, *web pack* and *web sort* to provide users with the flexibility to view sets of web documents in perspectives which may be more meaningful.

1 Introduction

Currently, web information may be discovered primarily by two mechanisms; browsers and search engines. This form of information access on the Web has a few shortcomings [6]. To resolve these limitations, we introduced Web Information Coupling System (WICS) [6], a database system for managing and manipulating coupled information extracted from the Web. WICS is one of the component of our web warehouse, called WHOWEDA (*W*are*h*ouse *o*f *We*b *Da*ta) [1,5,14]. In WICS, we materialize web information as *web tuples* and store it in a *web table*. We equip WICS with the basic capability to manipulate web tables and correlate additional, useful, related web information residing in the web tables [17]. Note that a web table is a collection of directed graphs (i.e., web tuples). The following example briefly illustrates query graph and web table, and provides the motivation for our work in this paper.

Example 1. Suppose a user Bill wish to find a list of drugs and their side effects on diseases from the WWW. We assume that there is a web site at `http://www.panacea.org/` which provides drug related information. Bill figured that there could be hyperlinks with anchor labels 'side effects' in `http://www.panacea.org/` that might be useful. To couple these related information from the WWW, Bill constructs a *coupling framework* (*query graph*) as shown in Figure 1[1].

* 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.

[1] In the figures, the boxes and directed lines correspond to *nodes* (Web document) and *links* respectively. Observe that some nodes and links have keywords imposed

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 68–80, 1999.

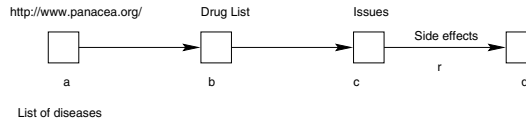


Fig. 1. Web schema (query graph) of 'Drugs' web table.

The global web coupling operator [6] can be applied to retrieve those set of related documents that match the coupling framework. Each set of inter-linked documents retrieved for the coupling framework is a directed graph (also called web tuples) and is materialized in web table called `Drugs`. A small portion of the web table is shown in Figure 2(a). Each web tuple in `Drugs` contains information about the side effects of a drug on a disease. ∎

The above approach of storing web information in a web table has the following shortcomings:

- It does not provide us with the ability to see the overall structure of the information captured in the web table. It is not possible for a user to visualize how one web tuple in a web table is related to another.
- The set of web tuples in a web table may contain duplicate web documents. For example, documents at `http://www.panacea.org/` (denoted by a_1) in all the web tuples in Figure 2(a) are identical. Similarly documents (denoted by b_1) in the first two web tuples are identical. Thus, a web table may contain duplicate documents and there is no mechanism to provide a *coalesced view* of the set of web tuples. A coalesced view allows a user to browse lesser number of directed connected graphs when locating information.
- It does not allow a user to group web tuples based on *related information content*, or *similar (or identical)* web sites. A user has to manually probe each web tuple to find these information. For example, the fifth, seventh and eighth web tuples in Figure 2(a) deal with the side effects of the drug `Beta Carotene`. However, these information cannot be grouped together in our web table.
- The set of web tuples are materialize in web tables. There is no other materialized representation of these web tables. For example, the collective view of these web tuples can be stored as a set of directed graphs having lesser number of nodes or links as compared to the original web table. This can optimize the query, storage and maintenance cost.

In this paper, we introduce some *data visualization operators* to resolve the above difficulties. These operators take as input a set of web tuples of a web table and provide a different view of the tuples as output. This gives users the flexibility to view documents in different perspectives that are more meaningful. Formally,

on them. These keywords express the content of the web document or the label of the hyperlink between the web documents.

we illustrate some data visualization operators and provide formal algorithm for these operators.

These operators provide different storage representation of web tuples which will help in optimizing the query, storage and maintenance cost. These different representations may also lead to inconsistency problems with respect to the original web table. Due to space limitations, we have not discussed the above problems in this paper.

2 Related Work

There has been considerable work in data model and query languages for the World Wide Web [11,12,13,16]. For example, Mendelzon, Mihaila and Milo [16] proposed a WebSQL query language based on a formal calculus for querying the WWW. The result of WebSQL query is a set of web tuples which are flattened immediately to linear tuples. Konopnicki and Shmueli [12] proposed a high level querying system called the W3QS for the WWW whereby users may specify content and structure queries on the WWW and maintain the results of queries as database views of the WWW. In W3QL, queries are always made to the WWW. Fiebig, Weiss and Moerkotte extended relational algebra to the World Wide Web by augmenting the algebra with new domains (data types) [11], and functions that apply to the domains. The extended model is known as RAW (Relational Algebra for the Web). Inspired by concepts in declarative logic, Lakshmanan, Sadri and Subramanian designed WebLog [13] to be a language for querying and restructuring web information. Other proposals, namely Lorel [2] and UnQL [9], aim at querying heterogeneous and semistructured information. However, none of these systems discuss data visualization operators similar to ours.

3 Background

Since our goal is to discuss data visualization operators in WICS, we use as a starting point the *Web Information Coupling Model* (WICM), designed as part of our Web Warehousing project (WHOWEDA). We describe WICM briefly, only to the extent necessary to understand the concept of data visualization operators. A complete description of the data model and its usefulness is given in [6,17].

3.1 Web Objects

It consists of a hierarchy of web objects. The fundamental objects are *Nodes* and *Links*. Nodes correspond to HTML or plain text documents and links correspond to hyper-links interconnecting the documents in the World Wide Web. We define a Node type and a Link type to refer to these two sets of distinct objects. These objects consist of a set of attributes as shown below:

```
Node = [url, title, format, size, date, text]
Link = [source-url, target-url, label, link-type]
```

For the `Node` object type, the attributes are the URL of a `Node` instance and its title, document format, size (in bytes), date of last modification, and textual contents. For the `Link` type, the attributes are the URL of the source document containing the hyperlink, the URL of the target document, the anchor or label of the link, and the type of the link. Hyperlinks in the WWW may be characterized into three types: *interior*, *local*, and *global* [16].

The next higher level of abstraction is a `web tuple`. A web tuple is a set of connected, directed graphs each consisting of a set of `nodes` and `links` which are instances of `Node` and `Link` respectively. A collection of web tuples is called a `web table`. If the table is materialized, we associate a `name` with the table. There is a *schema* (see next section) associated with every web table. A `web database` consists of a set of web schemas and a set of web tables.

3.2 Web Schema

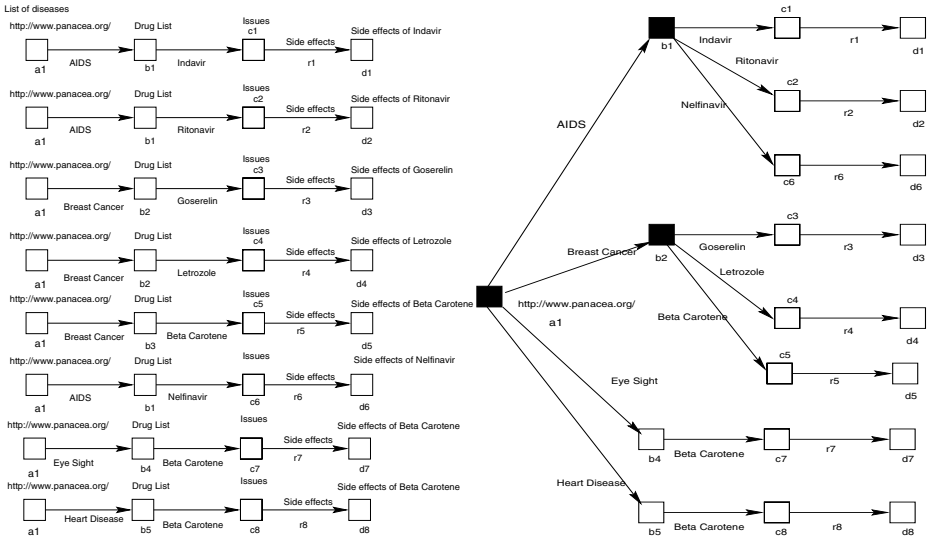
A web schema contains meta-information that binds a set of web tuples in a web table. Web tables are materialized results of web queries. In WICS, a user expresses a web query by describing a *query graph*.

When the query graph in Figure 1 is evaluated, a set of web tuples each *satisfying* the query graph is harnessed from the WWW. By collecting the tuples as a table, the query graph may be used as the table's schema to bind the tuples. Hence, the web schema of a table is the query graph that is used to derive the table. Formally, a web schema is an ordered 4-tuple $M = \langle X_n, X_\ell, C, P \rangle$ where X_n is a set of node variables, X_ℓ is a set of link variables, C is a set of connectivities (in Disjunctive Normal Form), and P is a set of predicates (in Disjunctive Normal Form).

Observe that some of the nodes and links in the figures have keywords imposed on them. To express these conditions, we introduced *node* and *link variables* in the query graph. Thus, in Figure 1 node d represents those web documents which contains the words 'side effects' in the text or title. In other words, variables denote arbitrary instances of `Node` or `Link`. There are two special variables: a node variable denoted by the symbol '`#`' and a link variable denoted by the symbol '`-`'. These two variables differ from the other variables in that they are never *bound* (these variables are not defined by the predicates of the schema).

Structural properties of web tuples are expressed by a set of *connectivities*. Formally, a connectivity k is an expression of the form: $x\langle\rho\rangle y$ where $x \in X_n$, $y \in X_n$, and ρ is a regular expression over X_ℓ. (The angle brackets around ρ are used for delimitation purposes only.) Thus, $x\langle\rho\rangle y$ describes a path or a set of possible paths between two nodes x and y.

The last schema component is the set of predicates P. Predicates provide a means to impose additional conditions on web information to be retrieved. Let p be a predicate. If x, y are node or link variables then the following are possible forms of predicates: $p(x) \equiv [x.\texttt{attribute CONTAINS "}A\texttt{"}]$ or $p(x) \equiv [x.\texttt{attribute EQUALS "}A\texttt{"}]$ and $p(x, y) \equiv [x.\texttt{attribute} = y.\texttt{attribute}]$. where `attribute` refers



(a) Web table Drugs. (b) Web nest operation.

Fig. 2.

to an attribute of `Node`, `Link` or `link_type`, A is a regular expression over the ASCII character set, x and y are *arguments* of p.

4 Web Data Visualization Operators

A query graph returns a set of web tuples which are stored as a web table. However, a user may wish to view these web tuples in different framework. In this section, we introduce some data visualization operators to add flexibility in viewing query results coupled from the WWW. Within this context, we propose an algorithm for each operator and illustrate each operator with an example.

4.1 Web Nest

Web tuples in a web table may be topically related. However, the relationship is not explicit from the way we store tuples in a web table. A visualization of this relationship gives an overview of the information space harnessed from the WWW and how these information are topically related. The `web nest` operator achieve this. It is an unary operator whose operand is a web table and whose output is a set of directed connected graphs condensed from the web table. Formally, let W be a web table with schema $M = \langle X_n, X_\ell, C, P \rangle$. Then, $G_n = \texttt{Nest}(W)$ where G_n is the set of condensed graphs.

```
Input: Web table W with schema M = ⟨X_n, X_ℓ, C, P⟩.
Output: A set of directed connected graphs G_n.

    Retrieve the first web tuple (denoted as t_i) from W;
    Store the tuple t_i in G_n;
    for each of the remaining web tuples (denoted as t_j) in W do {
        Z = 0;    for each directed connected graph in G_n {
            for each node x_j ∈ t_j {
                Get the URL of x_j;
                for each node x_i ∈ t_i {
                    if (URL of x_j = URL of x_i) {
                        Remove x_j;
                        Concatenate the out-bound and/or in-bound links
                        of x_j with the node x_i in G_n;
                        Z = Z + 1;
                    }
                    else {
                        Add x_j in G_n;
                        Retrieve the next node in t_i;
                    }
                }
                Retrieve the next node in t_j;
            }
            if (Z = 0) /* None of the nodes in t_j are identical to t_i */
                Add t_j in G_n as a seperate directed connected graph;
        }
    }
```

Fig. 3. Algorithm nest.

Algorithm Nest Algorithm nest is used to determine the topical relationship between a set of web tuples in a web table. It takes a web table as an input and produce a set of graphs as output. In WHOWEDA, a set of instances of a node variable may appear in more than one tuple in a web table. For example, in Figure 2(a) the node b_1 (an instance of node variable b) appears in first, second and sixth tuple. This indicates that the URLs of the instances of b in these web tuples are identical. Algorithm nest identifies multiple occurence of such identical nodes (by comparing the URLs of these nodes) in a set of web tuples and concatenate these tuples to one another over these identical nodes. Execution of this algorithm eliminates such replicated nodes in a web table. The formal description of the algorithm is given in Figure 3.

Example 2. Consider the web table in Figure 2(a). Application of web nest operator on the web table `Drugs` will result in a directed connected graph as shown in Figure 2(b) . Note that in Figure 2(b) none of the instances of node variables are replicated. The darkened boxes in the figure represents nodes over which web tuples are concatenated. ∎

4.2 Web Coalesce

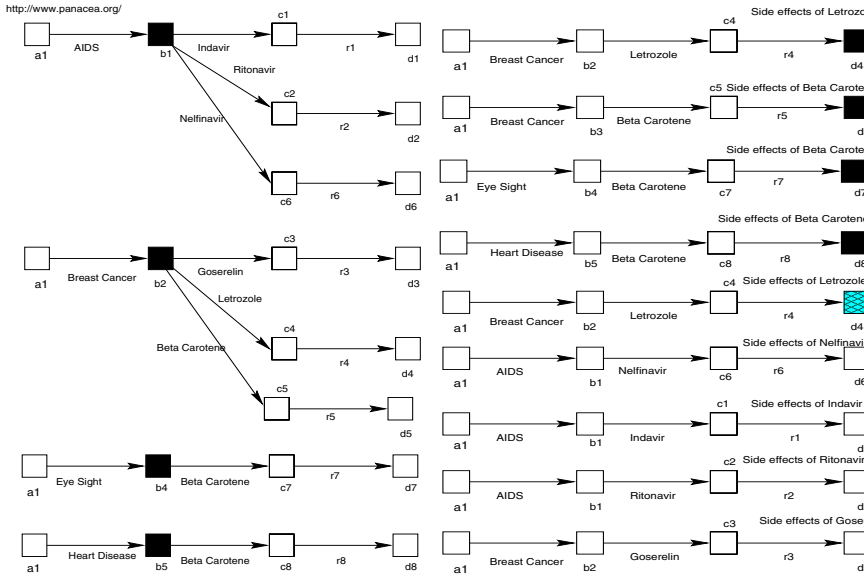
The set of web tuples in a web table may contain duplicate web documents. For example, documents denoted by b_1 in the first two web tuples in Figure 2(a) are identical. The number of duplicate nodes (documents) increases with the number of outgoing links of each nodes that satisfies the query graph. Thus, the size of a web table is proportional to the number of outgoing links satisfying the schema in each nodes. For reasonably small size of the web tables, browsing each web tuples for information of interest is a feasible option. However, browsing web tables of significant size may be tedious and inefficient way of locating information of interest. This problem may be minimized by coalescing duplicate nodes to reduce the size of the number of directed connected graphs in a web table. A coalesced web table results in a set of directed connected graphs and allows the user to browse lesser number of graphs (web tuples) compared to the original web table to locate desired information. To achieve this, we define the `web coalesce` operator. It takes a web table and a node variable as input and produce as ouput a set of directed connected graphs. It combines those web tuples in a web table which contains identical instances of the input node variable. Note that web coalesce is a specialization of web nest operation. Web nest coalesce web tuples by removing all duplicate nodes, whereas web coalesce operation coalesce a web table based on a node variable explicitly specified by a user. Formally, the web coalesce operation on node variable $x \in X_n$ is defined as $G_c = \texttt{Coalesce}(W, x)$.

Algorithm Coalesce The objective of Algorithm coalesce is similar to that of Algorithm nest. Analogous to Algorithm nest, it concatenates a set of web tuples in a web table. However, Algorithm coalesce only eliminates identical instances of a *specified* node variable from a set of web tuples as opposed to removal of all identical nodes in web nest operation. Algorithm coalesce takes as input a web table and a specified node variable based on which web coalesce operation is to be performed and produce a set of coalesced graph as output. First, it identifies multiple occurence of identical instances (same URL) of specified node variable by comparing the URLs of these nodes. Then, it concatenate the web tuples over these identical nodes to produce a set of coalesced graphs. The formal description of the algorithm is given in Figure 5.

Example 3. Consider the web table in Figure 2(a). Suppose Bill wish to coalesce the web tuples in `Drugs` on node variable b. This query will create a set of directed connected graphs as shown in Figure 4(a). The darkened boxes in the figure represents nodes over which web coalesce is performed. Note that each graph describes side effects of various drugs for a particular disease. ∎

4.3 Web Pack

The `web pack` operator groups the web tuples in a web tables based on *similar criteria*. It provides the flexibility to view web tuples containing *similar* nodes together. A user explicitly specifies the node variable and the criteria based on



(a) Web coalesce operation on b. (b) Web pack operation on d.

Fig. 4. Web coalesce and web pack operation.

which web pack operation is performed. The criteria based on which packing of web tuples can be performed is given below:

- `Host name:` Web tuples containing instances of specified node variable with identical host name are packed together.
- `Domain name:` Web tuples containing instances of specified node variable with identical domain names (.com, .edu, .org, .gov, .net, .mil etc) are grouped together.
- `Keyword content:` Packing may be performed by grouping web tuples whose instance of specified node variable contains user-specified keyword(s).

Note that in the case of packing based on keywords, the packed web table may contain duplicate web tuples since two different keywords may appear in the same node in a web tuple. Furthermore, tuples containing the keyword are displayed as top results and the remaining web tuples are pushed down in the packed web table. Formally, the web pack operation on node variable $x \in X_n$ is defined as $W_p = \texttt{Pack}(W, x, criteria)$ where W_p is the packed web table.

Algorithm Pack The objective of Algorithm pack is to group a set of web tuples (on a specified node variable) based on identical host name, domain name or keyword set. It takes as input a web table W with schema $M = \langle X_n, X_\ell, C, P \rangle$, a node variable $x \in X_n$ and packing condition P (P may be domain name, host name or keyword set) and produces as ouput a collection of packed web

Input: Web table W with schema $M = \langle X_n, X_\ell, C, P \rangle$, Node variable x such that $x \in X_n$.
Output: A set of directed connected graphs G_c.

```
Retrieve the first web tuple (denoted as t_i) from W;
Get the URL of the instance of the node variable x (denoted as u_i) in t_i;
Store the tuple t_i in G_c;
Remove t_i from W;
for each of the remaining web tuples (denoted as t_j) in W {
    Get the URL of the instance of node variable x (denoted as u_j) in t_j;
    for each directed connected graph in G_c {
        if (u_i = u_j) {
            Concatenate tuple t_j with t_i when the URL of nodes
            in t_j is identical to the corresponding URL of the nodes in t_i;
            Denote the concatenated directed graph in G_c as t_i;
            Remove the web tuple t_j from W;
        }
        else {
            Add t_j in G_c as a separate directed graph;
            Retrieve the next graph in G_c and denote it as t_i;
        }
    }
Retrieve the next web tuple from W;
}
```

Fig. 5. Algorithm coalesce.

tuples W_p based on the packing condition P. If the packing condition is host name or domain name then Algorithm pack retrieve the domain name or host name of instances of x (by checking the URLs of these nodes) in each web tuple and compare them to find if they are identical. Tuples with identical host or domain names are grouped together and stored in W_p. The remaining web tuples with distinct host or domain names are appended at the end of W_p. This displays of the group of web tuples with identical domain or host names as top results in W_p.

If the packing condition is a keyword set then Algorithm pack first checks the existence of the keyword(s) in each instance of x. Then, all the web tuples in W with identical keyword(s) (each keyword must be an element of the specified keyword set) are packed together and displayed as top results in W_p. Note that Algorithm pack may create duplicate web tuples when the packing condition is a keyword set. This is because different keywords may appear in the same document. The formal description of the algorithm is given in Figure 6.

Example 4. Continuing with the web table in Figure 2(a), suppose Bill wish to group all the web tuples in `Drugs` on node variable d which contains the keywords "Beta Carotene" or "Letrozole". The result of this query is shown in Figure 4(b). Note that the group of web tuples whose instance of node variable d contains the

Input: Web table W with schema $M = \langle X_n, X_\ell, C, P\rangle$, Node variable x such that $x \in X_n$,
Pack condition $P \in \{\texttt{host\_name}, \texttt{domain\_name}, \texttt{keyword\_set}\}$.
Output: Web table W_p.

```
(1)   Z = 0;
(2)   if (P = host_name or P = domain_name) {
(3)       Get the first web tuple (denoted as t1) in W;
(4)       Get the instance of node variable x in t1;
(5)       Get the host name (denoted as h1) or domain name (denoted as d1)
(6)       depending on P from the URL of the node;
(7)       for each of the remaining web tuples in W {
(8)           Get the instance of node variable x;
(9)           if (P = host_name)
(10)              Get the host name from the URL of the node;
(11)          else
(12)              Get the domain name;
(13)          if (host name of the node instance = h1 or domain name = d1) {
(14)              Append the web tuple at the end of Wp;
(15)              Remove the tuple from W;
(16)              Z = Z + 1;
(17)          }
(18)          else
(19)              ;
(20)      }
(21)      Append t1 at the end of Wp;
(22)      if (Z = 0)
(23)          Mark the tuple t1;
(24)      Remove t1 from W;
(25)      Repeat steps (1) to (24) for the remaining web tuples in W;
(26)      Move the set of marked web tuples at the end of Wp;
(27)  }
(28)  else {
(29)      for each keyword or combination of keywords in the keyword set {
(30)          for each web tuples in W {
(31)              Get the instance of node variable x;
(32)              Check if the keyword or set of keywords exist in the node;
(33)              if (keyword exists) {
(34)                  Append the web tuple at the end of Wp;
(35)                  Z = Z + 1;
(36)                  Mark the web tuple in W;
(37)              }
(38)              Get the next web tuple in W;
(39)          }
(40)      }
(41)  }
(42)  Get the set of web tuples which are unmarked in W;
(43)  Append these web tuples at the end of Wp;
```

Fig. 6. Algorithm pack.

Input: Web table W with schema $M = \langle X_n, X_\ell, C, P \rangle$,
Sort condition $S_c \in \{\texttt{node}, \texttt{local}, \texttt{global}, \texttt{interior}\}$,
Ordering type $O_t \in \{\texttt{asc}, \texttt{desc}\}$
Output: Web table W_s.

```
(1)   Z = 0;
(2)   if (S_c = node) {
(3)   for each web tuple in W {
(4)       Calculate total number of nodes N;
(5)       Store N;
(6)   }
(7)   else {
(8)       for each web tuple in W {
(9)           Z = 0;
(10)          for each link {
(11)              Get the link-type of the link;
(12)              if (S_c = local) {
(13)                  if (linktype is local)
(14)                      Z = Z + 1;
(15)                  else
(16)                      ;
(17)              }
(18)              else {
(19)                  if (S_c = global) {
(20)                      if (link-type is global)
(21)                          Z = Z + 1 ;
(22)                      else
(23)                          ;
(24)                  }
(25)                  else {
(26)                      if (link-type is interior)
(27)                          Z = Z + 1 ;
(28)                      else
(29)                          ;
(30)                  }
(31)              }
(32)          }
(33)          Store Z for each tuple;
(34)      }
(35)  }
(36)  if (O_t = asc)
(37)      Sort the tuples in ascending order based on N or Z;
(38)  else
(39)      Sort the tuples in descending order;
(40)  Store the sorted web tuples in W_s;
```

Fig. 7. Algorithm sort.

keyword "Beta Carotene" or "Letrozole" is displayed as top query results and remaining web tuples in `Drugs` are pushed down. The tuples whose instances of node variable d (the first four tuples) are filled with black are grouped together since they contain the keyword "Beta Carotene". The tuples whose instances of node variable d (fifth tuple) are filled with a pattern, are grouped together because they contain the keyword "Letrozole". Note that the node d_4 contains both the keywords. Thus, tuple containing d_4 is replicated (1st and 5th web tuples are identical) in the web table created after web pack operation. ∎

4.4 Web Sort

`Web sort` operator sort web tuples based on given *conditions*. The conditions to sort the web tuples in a web table in ascending or descending order are given below:

- Total number of nodes in each web tuple.
- Total number of specified link-type (local, global or interior) in each web tuple. This allows a user to measure the frequency of local, interior and global links in the web tuples returned in response to his query.

Web sort enables us to rearrange the tuples in ascending or descending order based on the number of occurrences of local, global or interior links in each tuple or total number of nodes in each tuple. Formally, web sort operation on a web table W with schema $M = \langle X_n, X_\ell, C, P\rangle$ is $W_s = \texttt{Sort}(\texttt{W}, \texttt{sort\_condition}, \texttt{order\_type})$ where *sort_condition* is the condition based on which sorting is performed and *order_type* specifies the ordering method (ascending or descending) of the sorted web tuples.

Algorithm Sort Algorithm sort rearranges the web tuples in ascending or descending order based on the sorting condition. It takes as input a web table W, sort condition S_c and ordering type O_t (ascending or descending order) and produces as output a sorted web table W_s based on the sorting condition. For each tuple in W, Algorithm sort calculates the total number of nodes or total number of local, global or interior links (depending on the sort condition S_c) and arranges these tuples in ascending or descending order. The formal description of the algorithm is given in Figure 7. Due to space limitations, we do not provide an example for web sort here. Refer to [8] for details.

5 Conclusion

In this paper, we have motivated the need for the data visualization operators and introduced some operators such as web nest, web coalesce, web pack and web sort that provides different flavors of visualizing web tables. Due to space limitations, we have not reported inverse operators like *web unnest*, *web expand*, *web unpack*. Refer to [8] for details. Currently, we are implementing these web operators as a part of a web query language.

References

1. http://www.cais.ntu.edu.sg:8000/~ whoweda/. 68
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Weiner. The Lorel Query Language for Semistructured Data. *Journal of Digital Libraries*, 1(1):68-88, April 1997. 70
3. G. Arocena, A. Mendelzon. WebOQL: Restructuring Documents, Databases and Webs. *Proceedings of ICDE 98*, Orlando, Florida, February 1998.
4. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Web Bags: Are They Useful in A Web Warehouse? *Proceedings of 5th International Conference of Foundation of Data Organization (FODO'98)*, Kobe, Japan, November 1998.
5. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Web Warehousing: Design and Issues. *Proceedings of International Workshop on Data Warehousing and Data Mining (DWDM'98) (in conjunction with ER'98)*, Singapore, 1998. 68
6. S. Bhowmick, W.-K. Ng, E.-P. Lim. Information Coupling in Web Databases. *Proceedings of the 17th International Conference on Conceptual Modelling (ER'98)*, Singapore, 1998. 68, 69, 70
7. S. S. Bhowmick, W.-K. Ng, E.-P. Lim, S. K. Madria. Join Processing in Web Databases. *Proceedings of the 9th International Conference on Database and Expert Systems Application (DEXA)*, Vienna, Austria, 1998.
8. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Data Visualization Operators in A Web Warehouse. *Technical Report*,CAIS-TR-98-20, Center for Advanced Information Systems, Nanyang Technological University, Singapore, 1998. 79
9. P. Buneman, S. Davidson, G. Hillebrand, D. Suciu. A query language and optimization techniques for unstructured data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Canada, June 1996. 70
10. M. Fernandez, D. Florescu, A. Levy, D. Suciu. A Query Language for a Web-Site Management Systems *SIGMOD Record*, 26(3), Sept, 1997.
11. T. Fiebig, J. Weiss, G. Moerkotte. RAW: A Relational Algebra for the Web. *Workshop on Management of Semistructured Data (PODS/SIGMOD'97)*, Tucson, Arizona, May 16, 1997. 70
12. D. Konopnicki, O. Shmueli. W3QS: A Query System for the World Wide Web. *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, 1995. 70
13. L.V.S. Lakshmanan, F. Sadri, I.N. Subramanian. A Declarative Language for Querying and Restructuring the Web *Proceedings of the Sixth International Workshop on Research Issues in Data Engineering*, February, 1996. 70
14. E.-P. Lim, W.-K. Ng, S. Bhowmick, F.-Q. Qin, X. Ye. A Data Warehousing System for Web Information.*Proceedings of 1st Asia Digital Library Workshop*, Hong Kong, August 5–7, 1998. 68
15. M. Liu, T. Guan, L. V. Saxton. Structured-Based Queries over the World Wide Web. *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, Singapore, 1998.
16. A. O. Mendelzon, G. A. Mihaila, T. Milo. Querying the World Wide Web. *Proceedings of the International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami, Florida, 70, 71
17. W. K. Ng, E.-P. Lim, C. T. Huang, S. Bhowmick, F. Q. Qin. Web Warehousing: An Algebra for Web Information. *Proceedings of IEEE International Conference on Advances in Digital Libraries (ADL'98)*, Santa Barbara, California, April 22–24, 1998. 68, 70

Recent Advances and Research Problems in Data Warehousing

Sunil Samtani[1], Mukesh Mohania[2], Vijay Kumar[1], and Yahiko Kambayashi[3]

[1] Dept. of Computer Science Telecommunications
University of Missouri-Kansas City, Kansas City, MO 64110, USA
ssamtani@cstp.umkc.edu
[2] Advanced Computing Research Centre
School of Computer and Information Science, University of South Australia
Mawson Lakes 5095 , Australia
mohania@cis.unisa.edu.au
[3] Department of Social Informatics, Kyoto University
Kyoto 606-8501, Japan
yahiko@kuis.kyoto-u.ac.jp

Abstract. In the recent years, the database community has witnessed the emergence of a new technology, namely *data warehousing*. A data warehouse is a global repository that stores pre-processed queries on data which resides in multiple, possibly heterogeneous, operational or legacy sources. The information stored in the data warehouse can be easily and efficiently accessed for making effective decisions. The On-Line Analytical Processing (OLAP) tools access data from the data warehouse for complex data analysis, such as multidimensional data analysis, and decision support activities. Current research has lead to new developments in all aspects of data warehousing, however, there are still a number of problems that need to be solved for making data warehousing effective. In this paper, we discuss recent developments in data warehouse modelling, view maintenance, and parallel query processing. A number of technical issues for exploratory research are presented and possible solutions are discussed.

1 Introduction

A data warehouse is an integrated repository that stores information which may originate from multiple, possibly heterogeneous operational or legacy data sources. The contents of a data warehouse may be a replica of a part of data from a source or they may be the results of *preprocessed* queries or both. This way of storing data provides a powerful tools to business organizations for making business decisions. The architecture of a data warehousing system allows a number of ways to integrate and query (such as *eager* or *in-advance*) information stored in it. This approach of integrating data from distributed data sources pays rich dividends when it translates into calculated decisions backed by sound analysis. Thus, the *Data warehousing* coupled with *On-Line Analytical Processing (OLAP)* enable business decision makers to creatively approach, analyze and

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 81–92, 1999.

understand business problems. The data warehouse system is used to provide solutions for business problems since it transforms operational data into strategic decision making information. The data warehouse stores summarized information over operational data. This summarized information is time-variant and provides effective answers to queries like "what are the supply patterns of 'toy' product in California in 1997 and how were they different from last year?" To extract this information from a distributed relational model, we would require to query multiple data sources and integrate the information at a particular point before presenting the answers to the user. This requires time and it is not *online*. In a data warehousing scenario, such queries find their answers in a central place, thus reducing the processing and management costs.

[Wid95] proposed a number of technical issues in data warehousing which needed the immediate attention of the research community. A lot of work has been done in many areas since then and new issues have come to the fore. [CD97] presents the state of art in data warehousing and OLAP, with emphasis on new requirements. The work presented serves as a very good overview and is an excellent survey in this area. However, there is a need to initiate a discussion on the impending research problems in the above mentioned fields. We need to delve into each research problem minutely rather than adopting a global approach. In this paper, we attempt to give new direction in some of these problems by providing insight for developing efficient solutions and by discussing some of the well known solutions.

The rest of the paper is organised as follows. In Section 2, we give an architecture of a data warehousing system. We discuss the multidimensional model and the implementation schemes in Section 3. In Section 4 we highlight the need for constraints in the multidimensional environment and its proposed uses. A promising challenge in data warehousing is how to maintain the materialized views. The view maintenance has branched into a number of sub-problems like self maintenance, consistency maintenance, update filtering and on-line view maintenance In Section 5, we investigate these sub-problems. We also discuss the issues of parallelism in data warehousing in Section 6. We mention the other areas of active research in Section 7. Section 8 concludes the paper with a discussion on possible future work.

2 Data Warehousing System

The data warehousing system architecture is illustrated in Figure 1. Warehouse data is derived from the data contained in operational data systems. These operational data sources are connected to the *wrappers/monitors*, whose main function is to select, transform and clean data. It also monitors changes to the source data and propagates them to the integrator. The integrator's job is to combine data selected from different data sources. It resolves conflicts among data and brings data in consistent form. After integration, data is propagated into warehouse storage. Many commercial analysis and development tools are

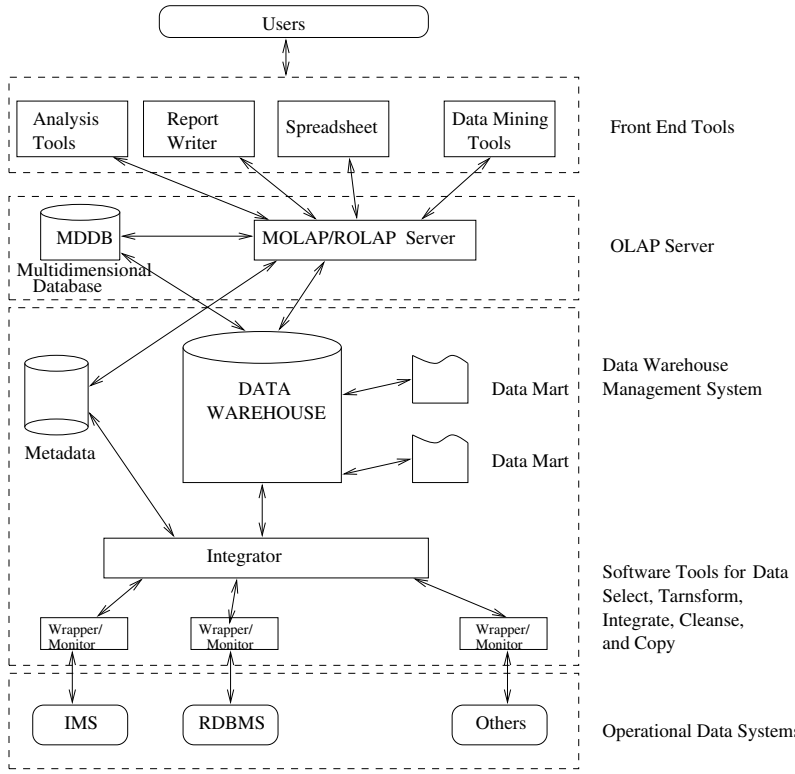


Fig. 1. A data warehousing architecture

available for data selecting, filtering, transforming, and loading. Examples include Platinum InfoRefiner and InfoPump, SAS Access, IBM Data joiner.

There are two approaches to creating the warehouse data, namely, bottom-up approach and top-down approach, respectively. In a bottom-up approach, the data is obtained from the primary sources based on the data warehouse applications and queries which are typically known in advance, and then data is selected, transformed, and integrated by data acquisition tools. In a top-down approach, the data is obtained from the primary sources whenever a query is posed. In this case, the warehouse system determines the primary data sources in order to answer the query. These two approaches are similar to *eager* and *lazy* approaches discussed in [Wid95]. The bottom-up approach is used in data warehousing because user queries can be answered immediately and data analysis can be done efficiently since data will always be available in the warehouse. Hence, this approach is feasible and improves the performance of the system. Another approach is a hybrid approach, which combines aspects of the bottom-up and top-down approaches. In this approach, some data is stored in a warehouse, and other data can be obtained from the primary sources on demand [HZ96].

The metadata contains the informational data about the creation, management, and usage of the data warehouse. It serves as a bridge between the users of the warehouse and the data contained in it. The warehouse data is accessed by OLAP server to present the same in a multidimensional way to the front end tools (such as analytical tools, report writer, spreadsheets, data mining tools) for analysis and informational purposes. Basically, OLAP server interprets client queries (the client interacts with front end tools and pass these queries to the OLAP server) and converts them into complex SQL queries (indeed extended SQL) required to access the warehouse data. It might also access the data from the primary sources if the client's queries need operational data. Finally, the OLAP server passes the multidimensional views of data to the front end tools, and these tools format the data according to the client's requirements.

3 The Multidimensional Data Model

The data models for designing traditional OLTP systems are not well-suited for modelling complex queries in data warehousing environment. The transactions in OLTP systems are made up of simple, pre-defined queries. In the data warehousing environments, queries tend to use joins on more tables, have a larger computation time and are ad-hoc in nature. This kind of processing environment warrants a new perspective to data modelling. The *multidimensional* data model i.e., the *data cube* turned out to be an adequate model that provides a way to aggregate facts along multiple attributes, called *dimensions*. Data is stored as *facts* and *dimensions* instead of rows and columns as in relational data model. Facts are numeric or factual data that represent a specific business activity and a dimension represent a single perspective on the data. Each dimension is described by a set of attributes.

The conceptual multidimensional data model can be physically realized in two ways, (1) by using trusted relational databases (*star schema/snowflake schema* [CD97]) or (2) by making use of specialised multidimensional databases. In this section we briefly describe multidimensional scheme.

Multidimensional Scheme

This scheme stores data in a matrix using array-based storage structure. Each cell in the array is formed by the intersection of all the dimensions, therefore, not all cells have a value. For example, not all database researchers publish in all journals. The multi-dimensional data set requires smaller data storage since the data is clustered compactly in the multidimensional array. The values of the dimensions need not be explicitly stored. The n-dimensional table schema is used to support multidimensional data representation which is described next.

***n*-dimensional Table Schema** An n-dimensional table schema is the fundamental structure of a multidimensional database which draws on terminology of the statistical databases. The attribute set associated with this schema is

of two kinds: *parameters* and *measures*. An n-dimensional table has a set of attributes R and a set of dimensions D associated with it. Each dimension is characterized by a distinct subset of attributes from R, called the parameters of that dimension. The attributes in R which are not parameters of any dimension are called the measure attributes. This approach is a very unique way of $flattening$ the data cube since the table structure is inherently multidimensional. The actual contents of the table are essentially orthogonal to the associated structure. Each *cell* of the data cube can be represented in an n-dimensional table as table entries. These table entries have to be extended by dimensions to interpret their meaning. The current literature on n-dimensional table however does not give an implementation of the MDDB which is different from the implementation suggested by the already existing schemas. This implementation breaks up the n-dimensional table into dimension tables and fact tables which snowballs into snowflake schema and traditional ROLAP. The challenge with the research community is to find mechanisms that translate this multidimensional table into a true multidimensional implementation. This would require us to look at new data structures for the implementation of multiple dimensions in one table. The relation in relational data model is a classic example of 0-dimensional table.

4 Constraints on the Multidimensional Databases

In a relational schema, we can define a number of integrity constraints in the conceptual design. These constraints can be broadly classified as key constraints, referential integrity constraints, not null constraint, relation-based check constraints, attribute-based check constraints and general assertions (business rules). These constraints can be easily translated into triggers that keep the relational database consistent at all times. This concept of defining constraints based on dependencies can be mapped to a multidimensional scenario.

The current literature on modelling multidimensional databases has not discussed the constraints on the data cube. In a relational model, the integrity and business constraints that are defined in the conceptual schema provide for efficient design, implementation and maintenance of the database. Taking a cue from the relational model, we need to identify and enumerate the constraints that exist in the multidimensional model. An exploratory research area would be to categorise the cube constraints into classes and compare them with the relational constraints. The constraints can be broadly classified into two categories: *intra-cube* constraints and *inter-cube* constraints.. The intra-cube constraints define constraints within a cube by exploiting the relationships that exist between the various attributes of a cube. The relationships between the various dimensions in a cube, the relationships between the dimensions and measure attributes in a cube, dimension attribute hierarchy and other cell characteristics are some of the key cube features that need to formalised as a set of intra-cube constraints. The inter-cube constraints define relationships between two or more cubes. There are various considerations in defining inter-cube constraints. Such constraints can be defined by considering the relationships between dimensions in different

cubes, the relationships between measures in different cubes, the relationships between measures in one cube and dimensions in the other cube and the overall relationship between two cubes, i.e., two cubes might *merge* into one, one cube might be a *subset* of the other cube, etc.

Using Constraints The cube constraints will facilitate the conceptual schema design of a data warehouse and allow us to provide triggers in a multidimensional scenario. In a relational database system, triggers play a very important role by enforcing the constraints and business rules in a very effective manner. The UPDATE, INSERT and DELETE triggers in the relational model have provided the robustness by allowing least manual intervention. The current data warehouse maintenance algorithms have neglected the role of triggers in designing effective view maintenance paradigms. The research community needs to envision the new types of triggers which would be needed for effective data warehousing. The constraint set will be of vital importance to this faculty.

The constraint set can also be used to solve the existing problems like view maintenance. In [MKK97,RSS96], the authors have proposed the view maintenance techniques based on the view expression tree that is used for evaluating the view as a query. The auxiliary relations are materialized for each node in the tree in addition to the materialized view. The updates to nodes in the expression tree are calculated and propagated in a bottom-up fashion. The update to each node in the tree is derived from the updates to its children nodes and the auxiliary relations materialized for the children and the node itself. If the constraints are defined on the view, then they can be broken down into smaller components (i.e. sub-constraints) and these sub-constraints can be pushed down into the expression tree to provide new efficient mechanisms for view maintenance and aggregate query optimization. That is, the constraints can be defined at each node in the tree. In this case, the updates to each node can be checked using these constraints before propagating them to the parent node. If some of the updates violate constraints, then there is no need to propagate these updates to the parent node. This method of decomposing the constraints and pushing them down along the edges of the expression tree can effectively reduce the communication and computation costs. The constraint set acts like a sieve which effectively filters the updates that do not affect the view.

Further, the constraint set will be an very effective tool that can be used in every aspect of data warehousing. The constraint set will enable us to define a complete algebra for the multidimensional data model which is independent of the underlying schema definitions. Also, the constraint set will allow evolution of new attributes in the multidimensional scenario. Some of these constraints can be used to map the multidimensional world into the relational world. We can view the data model as a soccer ball and the constraints can help us get an impression of that ball from a certain perspective. The constraint set will enable us to seek new, improved implementation mechanisms that are able to conserve the multidimensionality of data. Our first task is to define a constraint set on the multidimensional data model which is complete in any implementation scheme.

The constraints need to be formally defined and the precedence order for the constraints needs to be decided. We hope that the discussion we have initiated will spark off hectic activity in the research community.

5 View Maintenance

A *data warehouse* (*DW*) stores integrated information from multiple data sources in *materialized views* (*MV*) over the source data. The data sources (*DS*) may be heterogeneous, distributed and autonomous. When the data in any source (base data) changes, the *MVs* at the *DW* need to be updated accordingly. *The process of updating a materialized view in response to the changes in the underlying source data is called View Maintenance.* The view maintenance problem has evoked great interest in the past few years. This view maintenance in such a distributed environment gives rise to inconsistencies since there is a finite unpredictable amount of time required for (a) propagating changes from the *DS* to the *DW* and (b) computing view updates in response to these changes. Data consistency can be maintained at the data warehouse by performing the following steps:

- propagate changes from the data sources (ST_1 - current state of the data sources at the time of propagation of these changes) to the data warehouse to ensure that each view reflects a consistent state of the base data.
- compute view updates in response to these changes using the state ST_1 of the data sources.
- install the view updates at the data warehouse in the same order as the changes have occurred at the data sources.

The inconsistencies at the data warehouse occur since the changes that take place at the data sources are random and dynamic. Before the data warehouse is able to compute the view update for the *old* changes, the *new* changes change the state of the data sources from ST_1 to ST_2. This violates the consistency criterion that we have listed. Making the *MVs* at the data warehouse self-maintainable decimates the problem of inconsistencies by eliminating the finite unpredictable time required to query the data source for computing the view updates. In the next subsection, we describe self-maintenance of materialized views at the data warehouse.

Self-Maintenance Consider a materialized view MV at the data warehouse defined over a set of base relations $R = \{R_1, R_2, ..., R_n\}$. MV stores a preprocessed query at the data warehouse. The set of base relations R may reside in one data source or in multiple, heterogenous data sources. A change ΔR_i made to the relation R_i might affect MV. MV is defined to be self-maintainable if a change ΔMV in MV, in response to the change ΔR_i can be computed using only the MV and the update ΔR_i. But the data warehouse might need some additional information from other relations in the set R residing in one

or more data sources to compute the view update ΔMV. Since the underlying data sources are decoupled from the data warehouse, this requires a finite computation time. Also the random changes at the data sources can give rise to inconsistencies at the data warehouse. Some data sources may not support full database functionalities and querying such sources to compute the view updates might be a cumbersome, even an impossible task. Because of these problems, the preprocessed query that is materialized at the warehouse needs to be maintained without access to the base relations.

One of the approaches is to replicate all base data in its entirety at the data warehouse so that maintenance of the MV becomes local to the data warehouse. [GM95,Küc91,GMS93]. Although this approach guarantees self-maintainability at the warehouse, it creates new problems. As more and more data is added to the warehouse, it increases the space complexity and gives rise to information redundancy which might lead to inconsistencies. This approach also overlooks the point that the base tuples might be present in the view itself, so the view instance, the base update and a subset of the base relations might be sufficient to achieve self-maintainability in the case of SPJ (Select-Project-Join) views [Huy97]. *But how can the subset of the base relations that is needed to compute the view updates be stored at DW?* This question was addressed in [QGMW96], which defines a set of minimal *auxiliary views (AVs)* to materialize that are sufficient to make a view self-maintainable. Although materializing auxiliary views at the DW was a novel concept, the minimality of auxiliary views defined was still questionable since the MV instance was never exploited for self-maintenance. Most of the current approaches maintain the MVs separately from each other using a separate *view manager* for each view and such approaches fail to recognize that these views can be maintained together by identifying the set of related materialized views. This issue of multiple-view self-maintenance was addressed for the first time in [Huy97].

In some approaches to multiple-view self-maintenance, a set of auxiliary views (AV) are stored at the data warehouse along with the set of materialized views (MV) such that together $MV \cup AV$ is self-maintainable. The research challenge lies in finding the most *economical* AVs in terms of space complexity and computational costs. The view self maintenance is still an active research problem. It is not always feasible to provide self-maintainability of the views at the data warehouse. When the cost of providing self-maintainability exceeds the cost of querying data sources for computing view updates, it is profitable to allow querying of data sources instead.

Update Filtering The changes that take place in the source data need to be reflected at the data warehouse. Some changes may create view updates that need to installed at the data warehouse; some changes leave the views at the data warehouse unchanged. If we are able to detect at the data sources that certain changes are guaranteed to leave the views unchanged, we need not propagate these changes to the data warehouse. This would require checking of distributed integrity constraints at a single site. As many changes as possible

can be filtered at the sources and only the changes that result in view updates may be propagated to the warehouse. The update filtering will reduce the size of the maintenance transactions at the data warehouse, thus minimizing the time required to make the data warehouse consistent with the data sources. The side effect of update filtering is that we need to make our data sources (and the wrapper/monitor) components more intelligent. They need to know about their participation in the data warehouse and the data warehouse configuration so that the updates can be checked against the constraint set before propagating them. To be able to realize this, the data sources cannot be decoupled from the data warehouse anymore. This would give rise to new problems like configuration management i.e., if there is a change in the schema at any data source or at the data warehouse, all the participating entities need to be informed of this change so that they can modify the constraint set to reflect this change. The view maintenance strategies would now be based on the constraint set and any change to the constraint set would warrant a change in the existing view maintenance transaction.

On-Line View Maintenance Warehouse view maintenance can be either done *incrementally* or by queueing a large number of updates at the data sources to be propagated as a *batch* update from the data sources to the data warehouse. In current commercial systems, a batch update is periodically sent to the data warehouse and view updates are computed and installed. This transaction is called the *maintenance transaction*. A user typically issues read-only queries at the data warehouse and a long-running sequence of user queries is called a *reader session*. The batch maintenance transaction is typically large and blocks the reader sessions. This makes the data warehouse $offline$ for the duration of the maintenance transaction. The maintenance transaction typically runs at night. With the advent of the internet and global users, this scheme will have to give way. The $24 - hour$ shop concept is what most companies are striving for and the data warehouse to be online 24 hours to allow the company to be competitive in its strategies. Incremental view maintenance which updates the data warehouse instantaneously in response to every change at the data source is expensive and gives rise to inconsistent results during the same reader session. An update from the data source will change the results a user might see over a sequence of queries. We need to get around these problems. [QW97] discusses a possible approach to this problem by maintaining two versions of each tuple at the data warehouse simultaneously so that the reader sessions and the maintenance transactions do not block each other. A possible solution may need the integration with self maintenance techniques, where auxiliary views can be used to answer queries during maintenance transactions.

6 Parallelism in Data Warehousing

In this paper, we have discussed every possible technique to evaluate OLAP queries faster. All these techniques are restricted in their ways. The precompu-

tation strategy needs to anticipate queries so that it can materialize them in the data warehouse. OLAP queries are of adhoc nature and restricts precomputing to the imagination of the designer. The indexing schemes we have discussed provide faster access to the data stored in the warehouse, but does not reduce the size of tables stored at the data warehouse. One can say that the strategies presented provide faster query processing, but they need not be fast enough as perceived by the user. One mechanism that is recently being exploited by vendors to compute queries quickly is to execute the query in parallel by partitioning data among a set of processors. This can potentially achieve *linear speedup* and can significantly improve query response times. However to exploit this technology we need to address a few key issues like parallel data placement and parallel join.

The existing data warehouse schema design strategies is not catered to parallel data placement, although the star schema suggests implicitly the need of partitioning the data into a set of dimension tables and fact tables. Many DBMS vendors claim to support parallel data warehousing to various degrees. Most of these products, however, do not use the dimensionality of data that exists in a data warehouse. The effective use of parallel processing in this environment can be achieved only if we are able to find innovative techniques for parallel data placement using the underlying properties of data in the warehouse. [DMT98] use the data indexing strategy to provide efficient data partitioning and parallel resource utilization. Effective algorithms that split the data among n parallel processors and perform parallel join operation have been illustrated. We need to take a cue from this work and find more effective solutions.

7 Other Research Challenges

The research challenges discussed so far are directly concerned with the problem of faster data integration. There are other challenges which are a result of the data warehousing concept. Here we mention a few such challenges which are currently active. One such problem arises due to the changing user requirements. As the requirements of the user changes, the view definitions at the data warehouse change dynamically. We need to develop view adaptation techniques that do not need the entire recomputation of the materialized view at every change. In [GMR95,MD96,Moh97], the authors have presented the view adaptation techniques for a data warehouse that recompute only parts of the view (if any) which cannot be derived from the existing materialized views. These techniques are applicable for SPJ queries. However, there is a need to devise adaptation algorithms for aggregated queries.

One issue which has seen little investigation is the fragmentation of a multidimensional database. Fragmentation plays an important role in the design of a distributed database system, it enables the definition of appropriate units of distribution which enhance query performance by enabling concurrent and parallel execution of queries. The requirement for parallel processing of OLAP operations to achieve quick response times is also important [DMT98]. This can be

acheived by fragmenting multidimensional database into number of fragments. We identify two types of possible fragmentation; (1) *slice 'n dice* that selects subsets of data by 'cutting up' the multidimensional hypercube representation of the global data into sub-cubes. This strategy is currently employed by vendors of multidimensional databases that support fragmentation; for example the Essbase OLAP Server Partitioning option. (2) *severing* that divides the global data by removing dimensions from a hypercube. This method may be used to divide a cube into two cubes with different dimensions from each other. Operations performed during severing ensure that the original cube may be reconstructed from the two severed cubes.

Another interesting problem is the problem of data expiry in the warehouse. Data that is materialized in the warehouse may not ne needed after a point. We need to devise efficient schemes that determine the data expiry point and do effective garbage collection by removing the unnecessary tuples.

8 Conclusions

This paper brings together under one roof all the techniques of integrating data quickly to help effective decision making. The data warehousing phenomenon, which has grown from strength to strength in the last few years presents new challenges everyday as we find new applications where warehousing can play a crucial role. In this paper, we have discussed the recent advances and research problems in three different areas, namely, data warehouse modeling, view maintenance, parallel query processing. Since the four areas discussed in this paper directly contribute in a large way to the *warehousing* phenomenon, there is a need to integrate the research work that is done in these different areas under one umbrella. This paper serves as a stepping stone in this direction.

Further, we have exploited the multidimensionality of data in the warehouse to introduce the concept of constraints on the data cube. We highlight the usefulness of the constraint set in every aspect of data warehousing, that is, design, implementation and maintenance. This paper also underlines the need for dynamic self-maintainability of materialized views and discusses the issues involved. In this paper, new areas for exploratory research have been presented and we hope that the issues presented will invoke keen interest from the data warehousing community.

References

CD97. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. In *ACM SIGMOD Record*, volume 26, pages 65–74. 1997. 82, 84

DMT98. A. Datta, B. Moon, and H. Thomas. A case for parallelism in data warehousing and olap. Technical report, Dept. of MIS, University of Arizona, Tucson, AZ URL: http://loochi.bpa.arizona.edu, 1998. 90

GM95. A. Gupta and I. S. Mumick. Maintenance of materialized views: problems, techniques, and applications. *IEEE Data Engineering Bulletin, Special Issue on Materialized Views and Warehousing*, 18(2), 1995. 88

GMR95. A. Gupta, I.S. Mumick, and K.A. Ross. Adapting materialized views after redefinitions. In *Proc. ACM SIGMOD International Conference on Management of Data, San Jose, USA*, 1995. 90

GMS93. A. Gupta, I. S. Mumick, and V. S. Subrahmanian. Maintaining views incrementally. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 157–166, 1993. 88

Huy97. N. Huyn. Multiple view self-maintenance in data warehousing environments. In *To Appear in Proc. Int'l Conf. on Very Large Databases*, 1997. 88

HZ96. R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. ACM SIGMOD Conf. On Management of Data*, pages 481–492, 1996. 83

Küc91. V. Küchenhoff. On the efficient computation of the difference between consecutive database states. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *Proc. Second Int. Conf. on Deductive Object-Oriented Databases*, volume 566 of *Lecture Notes in Computer Science, Springer-Verlag*, pages 478–502. Springer-Verlag, 1991. 88

MD96. M. Mohania and G. Dong. Algorithms for adapting materialized views in data warehouses. In *Proc. of International Symposium on Cooperative Database Systems for Advanced Applications, Kyoto, Japan*, pages 62–69, 1996. 90

MKK97. M. Mohania, S. Konomi, and Y. Kambayashi. Incremental maintenance of materialized views. In *Proc. of 8^{th} International Conference on Database and Expert Systems Applications (DEXA '97)*. Springer-Verlag, 1997. 86

Moh97. M. Mohania. Avoiding re-computation: View adaptation in data warehouses. In *Proc. of 8^{th} International Database Workshop, Hong Kong*, pages 151–165, 1997. 90

QGMW96. Dallan Quass, Ashish Gupta, Inderpal Singh Mumick, and Jennifer Widom. Making views self-maintainable for data warehousing. In *Proc. of International Conference on Parallel and Database Information Systems*, 1996. 88

QW97. D. Quass and J. Widom. On-line warehouse view maintenance for batch updates. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1997. 89

RSS96. K.A. Ross, D. Srivastava, and Sudarshan S. Materialized view maintenance and integrity constraint checking: Trading space for time. In *Proc. ACM SIGMOD International Conference on Management of Data, Montreal, Canada*, 1996. 86

Wid95. Jennifer Widom. Research problems in data warehousing. In *Proc. Fourth Intl. Conference on Information and Knowledge Management*, 1995. 82, 83

Web Warehousing: Design and Issues*

S. S. Bhowmick, S. K. Madria, W.-K. Ng, and E.-P. Lim

Center for Advanced Information Systems, School of Applied Science
Nanyang Technological University, Singapore 639798, Singapore
{sourav,askumar,wkn,aseplim}@cais.ntu.edu.sg

Abstract. The World Wide Web is a distributed global information resource. It contains a large amount of information that have been placed on the web independently by different organizations and thus, related information may appear across different web sites. To manage and access heterogeneous information on WWW, we have started a project of building a web warehouse, called WHOWEDA (*W*are*h*ouse *o*f *We*b *Da*ta). Currently, our work on building a web warehousing system has focused on building a data model and designing a web algebra. In this paper, we discuss design and research issues in a web warehousing system. The issues include are designing algebraic operators for web information access and manipulation, web data visualization and web knowledge discovery. These issues will not only overcome the limitations of available search engines but also provide powerful and friendly query mechanisms for retrieving useful information and knowledge discovery from a web warehouse.

1 Introduction

Most users obtain WWW information using a combination of search engines and browsers. However, these two types of retrieval mechanisms do not necessarily address all of a user's information needs and have the following shortcomings:

- Web browsers fully exploit hyperlinks among web pages, however, search engines have so far made little progress in exploiting link information. Not only do most search engines fail to support queries on the web utilizing link information, they also fail to return link information as part of a query's result.
- The search is limited to string matching. Numeric comparisons, as in conventional databases, cannot be done.
- Queries are evaluated on the index data rather than on the original and up-to-date data.
- The accuracy of results is low as there is an almost unavoidable repetition of information and existence of non-relevant data in the results.

* 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.

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 93–105, 1999.

- From the query's result returned by the search engines, a user may wish to couple a set of related Web documents together for reference. Presently, he may only do so manually by visiting and downloading these documents as files on the user's hard disk. However, this method is tedious, and it does not allow a user to retain the *coupling framework*.
- The set of downloaded documents can be refreshed (or updated) only by repeating the above procedure.
- If a user has successfully coupled a set of Web documents together, he may wish to know if there are other Web documents satisfying the same coupling framework. Presently, the only way is to request the same or other search engines for further Web documents and probe these documents manually.
- Over a period of time, there will be a number of coupled collections of Web documents created by the user. As each of these collections exists simply as a set of files on the user's system, there is no convenient way to manage and infer further useful information from them.

To overcome the limitations explained above and provide the user with a powerful and friendly query mechanism for accessing information on the web, the critical problem is to find the effective ways to build web data models of the information of interest, and to provide a mechanism to manipulate these information to garner additional useful information. The key objective of our web warehousing project, called WHOWEDA (*W*are*h*ouse *o*f *We*b *Da*ta), at the Centre for Advanced Information Systems in Nanyang Technological University, Singapore is to design and implement a web warehouse that materializes and manages useful information from the Web [22]. To meet the warehousing objective, we materialize coupled web information in the form of *web tuples* and store them in *web tables*. We define a set of web operators with web semantics to equip the warehouse with the basic capabilities to manipulate web tables and couple additional, useful, related web information residing in the web tables [9,22]. These operators include web algebraic operations such as web select, web join, web union, web intersection, and so on [22]. In this paper, we present an overview of a web warehouse and highlight new research directions in some of the important areas of building a web warehousing system.

1.1 Related Work

There has been considerable work in data model and query languages for the World Wide Web [13,15,16,20]. For example, Mendelzon, Mihaila and Milo [20] proposed a WebSQL query language based on a formal calculus for querying the WWW. The result of a WebSQL query is a set of web tuples which are flattened immediately to linear tuples. Konopnicki and Shmueli [15] proposed a high level querying system called the W3QS for the WWW whereby users may specify content and structure queries on the WWW and maintain the results of queries as database views of the WWW. In W3QL, queries are always made to the WWW. Fiebig, Weiss and Moerkotte extended relational algebra to the World Wide Web by augmenting the algebra with new domains (data types) [13], and functions

that apply to the domains. The extended model is known as RAW (Relational Algebra for the Web). Inspired by concepts in declarative logic, Lakshmanan, Sadri and Subramanian designed WebLog [16] to be a language for querying and restructuring web information. Other proposals, namely Lorel [2] and UnQL [11], aim at querying heterogeneous and semistructured information. These languages adopt a lightweight data model (based on labeled graphs) to represent data, and concentrate on the development of powerful query languages for these structures.

2 Web Information Coupling Model

In Web Information Coupling System (WICS) [9], we materialize web information as web tuples stored in web tables. Each web table is associated with a web schema. We equip the WICS with the basic capability to manipulate web tables and correlate additional useful and related web information residing in the web tables [22]. We proposed a Web Information Coupling Model (WICM) which describe web objects, web schema and a web algebra for retrieving information from the web and manipulating these information to derive additional information.

2.1 Web Objects

It consists of a hierarchy of web objects. The fundamental objects are *Nodes* and *Links*. Nodes correspond to HTML or plain text documents and links correspond to hyper-links interconnecting the documents in the World Wide Web. We define a Node type and a Link type to refer to these two sets of distinct objects. These objects consist of a set of attributes as shown below:

```
Node = [url, title, format, size, date, text]
Link = [source-url, target-url, label, link-type]
```

For the `Node` object type, the attributes are the URL of a `Node` instance and its title, document format, size (in bytes), date of last modification, and textual contents. For the `Link` type, the attributes are the URL of the source document containing the hyperlink, the URL of the target document, the anchor or label of the link, and the type of the link. Hyperlinks in the WWW may be characterized into three types: *interior*, *local*, and *global* [20].

The next higher level of abstraction is a `web tuple`. A web tuple is a set of connected, directed graphs each consisting of a set of `nodes` and `links` which are instances of `Node` and `Link` respectively. A collection of web tuples is called a `web table`. If the table is materialized, we associate a `name` with the table. There is a *schema* (see next section) associated with every web table. A `web database` consists of a set of web schemas and a set of web tables.

2.2 Web Schema

A web schema contains meta-information that binds a set of web tuples in a web table. Web tables are materialized results of web queries. In WICS, a user expresses a web query by describing a *query graph*.

When the query graph is evaluated, a set of web tuples each *satisfying* the query graph is harnessed from the WWW. By collecting the tuples as a table, the query graph may be used as the table's schema to bind the tuples. Hence, the web schema of a table is the query graph that is used to derive the table. Formally, a web schema is an ordered 4-tuple $M = \langle X_n, X_\ell, C, P \rangle$ where X_n is a set of node variables, X_ℓ is a set of link variables, C is a set of connectivities (in Disjunctive Normal Form), and P is a set of predicates (in Disjunctive Normal Form).

Observe that some of the nodes and links in the figures have keywords imposed on them. To express these conditions, we introduced *node* and *link variables* in the query graph. Thus, in Figure **??** node d represents those web documents which contains the words 'side effects' in the text or title. In other words, variables denote arbitrary instances of `Node` or `Link`. There are two special variables: a node variable denoted by the symbol '`#`' and a link variable denoted by the symbol '`-`'. These two variables differ from the other variables in that they are never *bound* (these variables are not defined by the predicates of the schema).

Structural properties of web tuples are expressed by a set of *connectivities*. Formally, a connectivity k is an expression of the form: $x\langle\rho\rangle y$ where $x \in X_n$, $y \in X_n$, and ρ is a regular expression over X_ℓ. (The angle brackets around ρ are used for delimitation purposes only.) Thus, $x\langle\rho\rangle y$ describes a path or a set of possible paths between two nodes x and y.

The last schema component is the set of predicates P. Predicates provide a means to impose additional conditions on web information to be retrieved. Let p be a predicate. If x, y are node or link variables then the following are possible forms of predicates: $p(x) \equiv [x.$`attribute CONTAINS "`A`"`$]$ or $p(x) \equiv [x.$`attribute EQUALS "`A`"`$]$ and $p(x, y) \equiv [x.$`attribute` $= y.$`attribute`$]$. where `attribute` refers to an attribute of `Node`, `Link` or `link_type`, A is a regular expression over the ASCII character set, x and y are *arguments* of p.

3 Web Algebra

The web algebra provides a formal foundation for data representation and manipulation for the web warehouse. It supports structured and topological query with sets of keywords specified on multiple nodes and on hyperlinks among the nodes. The user query is a graph-like structure and it is used to match the portions of WWW satisfying the conditions. The query result is a set of graphs called web tuples. We then define a set of web operators with web semantics to manipulate web tuples stored in a web table. The basic algebraic operators include global and local web coupling, web select, web join, web intersection, web union, etc. These operators are implemented as a part of our web query language. Briefly, these operators are discussed below. More details can be found in [5,9,22].

3.1 Information Access Operator

Global Web Coupling Global coupling enables a user to retrieve a set of collections of inter-related documents satisfying a web schema or coupling framework, regardless of the locations of the documents in the Web. To initiate global coupling, the user specifies the coupling framework in the form of a *query graph*. The coupling is performed by the WIC system and is transparent to the user. Formally, the global web coupling operator Γ takes in a query (expressed as a schema M) and extracts a set of web tuples from the WWW satisfying the schema. Let W_g be the resultant table, then $W_g = \Gamma(M)$. Each web tuple matches a portion of the WWW satisfying the conditions described in the schema. These related set of web tuples are coupled together and stored in a web table. Each web tuple in the web table is structurally identical to the schema of the table. The formal details appear in [9].

3.2 Information Manipulation Operators

Web Select The `select` operation on a web table extract web tuples from a web table satisfying certain conditions. However, since the schema of web tables is more complex than that of relational tables, selection conditions have to be expressed as predicates on node and link variables, as well as connectivities of web tuples. The `web select` operation augments the schema of web tables by incorporating new conditions into the schema. Thus, it is different from its relational counterpart.

Let W be a web table with schema $M = \langle X_n, X_\ell, C, P\rangle$. Selection condition(s) on that table is denoted by another schema $M_s = \langle X_{s,n}, X_{s,\ell}, C_s, P_s\rangle$ where C_s contains the selection criteria on connectivities, and P_s contains predicates on node and link variables in $X_{s,n}$ and $X_{s,\ell}$ respectively. Formally, we define web select as follows: $W_s = \sigma_{M_s}(W)$ where σ is the select operator.

Web Project The web project operation on a web table extract portions of a web tuple satisfying certain conditions. However, since the schema of web tables is more complex than that of relational tables, *projection conditions* have to be expressed as node and link variables and/or connectivities between the node variables. The web project operation reduces the number of node and link variables in the original schema and the constraints over these variables. For more details, refer to [5].

Given a web table W with schema $M = \langle X_n, X_\ell, C, P\rangle$, a web projection on W computes a new web table W' with schema $M_p = \langle X_{n_p}, X_{\ell_p}, C_p, P_p\rangle$. The components of M_p depends on the project conditions. Formally, we define web project as follows: $W' = \pi_{\langle project\_condition(s)\rangle}(W)$ where π is the symbol for project operation.

A user may explicitly specify any one of the conditions or any combination of the three conditions discussed below to initiate a web project operation.

- **Set of node variables:** To project a set of node variables from the web table.
- **Start-node variable and end-node variable:** To project all the instances of node variables between two node variables.
- **Node variable and depth of links:** To restrict the set of nodes to be projected within a limited number of links starting from the specified node variable.

Web Cartesian Product A web cartesian product, denoted by $\times$, is a binary operation that combines two web tables by concatenating a web tuple of one web table with a web tuple of other. If W_i and W_j are web tables with n and m web tuples respectively, the resulting web table W created by web cartesian product consists of $n \times m$ web tuples.

Local Web Coupling Given two web tables, local coupling is initiated explicitly by specifying a pair(s) of web documents (*coupling node variables*) and a set of keyword(s) to relate them. The result of local web coupling is a web table consisting of a set of collections of inter-related Web documents from the two input tables. To elaborate further, let W_i and W_j be two web tables with schemas $M_i = \langle X_{i,n}, X_{i,\ell}, C_i, P_i \rangle$ and $M_j = \langle X_{j,n}, X_{j,\ell}, C_j, P_j \rangle$ respectively. Let w_i and w_j be two web tuples from W_i and W_j, and $n_c(w_i)$ and $n_c(w_j)$ are instances of node variables n_{c_i} and n_{c_j} respectively. Suppose documents at `http://www.virtualdisease.com/cancer/index.html` (represented by node $n_c(w_i)$) and `http://www.virtualdrug.com/cancerdrugs/index.html` (represented by node $n_c(w_j)$) contain information related to cancer and appears in w_i and w_j respectively. Tuples w_i and w_j are *coupling-compatible locally* on $n_c(w_i)$ and $n_c(w_j)$ since they both contain similar information (information related to cancer).

We express local web coupling between two web tables as follows:

$$W = W_i \otimes_{(\{\langle node\_pair \rangle, \langle keyword(s) \rangle\})} W_j$$

where W_i and W_j are the two web tables participating in the coupling operation and W is the coupled web table satisfying a schema $M = \langle X_n, X_\ell, C, P \rangle$. In this case, $\langle node\_pair \rangle$ specifies a pair of coupling node variables in the web table W_i and W_j, and $\langle keyword(s) \rangle$ specifies a list of keyword(s) on which the similarity between the coupling node variable pair is evaluated.

Web Join The web join operator combines two web tables by *concatenating* a web tuple of one table with a web tuple of other table whenever there exists *joinable nodes*. Let W_i and W_j be two web tables with schemas $M_i = \langle X_{i,n}, X_{i,\ell}, C_{i,p}, P_i \rangle$ and $M_j = \langle X_{j,n}, X_{j,\ell}, C_j, P_j \rangle$ respectively. Then W_i and W_j are *joinable* if and only if there exist at least one node variable in M_i and in M_j which refers to identical (having the same URL) node or web document.

Consider the following predicates of the node variables c and z where $c \in X_{i,n}$ and $z \in X_{j,n}$:

$$p_{i_4}(c) \equiv [c.\texttt{url EQUALS "http://www.singapore.com/area/"}],$$
$$p_{j_4}(z) \equiv [z.\texttt{url EQUALS "http://www.singapore.com/area/"}]$$

Since the node variables c and z of M_i and M_j respectively refers to the same web document at URL '`http://www.singapore.com/area/`', the web tables W_i and W_j are joinable. The joinable nodes are c and z.

Formally, we define $W = W_i \bowtie W_j$ is a set of web tuples satisfying schema $M = \langle X_n, X_\ell, C, P \rangle$ where X_n is the set of node variables appearing in P, X_ℓ is the set of link variables appearing in P, C and P are obtained from M_i and M_j. We discussed web join operator in detail in [10].

Schema Tightness In a web warehouse, a user expresses a web query by describing a query graph or coupling framework. The query graph is used as the schema of the web table to bind the web tuples. In reality, it is unrealistic to assume from the (naive) user complete knowledge of the structure of the query graph. The user may express some incomplete graph structure based on the partial knowledge. Thus, such query graphs, if used as schema of the web table, may contain unbound nodes and links. Furthermore, a web schema serves two important purposes: First, it enables users to understand the structure of the web table and form meaningful queries over it. Second, a query processor relies on the schema to devise efficient plans for computing query results. Both the tasks become significantly harder when the schema contains unbound nodes and links.

To address these challenges, we design and implement a web operator called *schema tightness operator*. The schema tightness operator takes as input a web table containing unbound nodes and links and web schema and *tighten* the web schema of the web table by imposing constraints on the unbound nodes and links in the web table.

4 Web Data Visualization

A query graph returns a set of web tuples which are stored in a web table. However, a user may wish to view these web tuples in different framework. Here we present some data visualization operators to add flexibility in viewing query results coupled from the WWW and to generate some additional useful information.

4.1 Web Ranking Operators

Presently, in the WICM we have web operators to manipulate information from the WWW globally and locally. A crucial problem that a WIC system faces is extracting *hot tuples* for a user query (the most relevant web tuples). This problem is challenging because hot tuples might not be displayed at the beginning of

the web table. We are developing two web ranking operators; *global ranking* and *local ranking* operators to rank web tuples generated by global and local web coupling respectively. The ranking operators are based on the following factors:

- Number of occurrence of a keyword in a document.
- Location of occurrence of a keyword in a document, i.e.,whether the keyword occurs in the title, text or anchor of a document.
- Overlap between all pairs of web documents, i.e., if A, B and C are documents in three web tuples considered relevant (in that order) to a user query, then we believe that the "interestingness" of B is lower than C if B overlaps with A significantly, while C is a distinct result. Thus, our ranking function will rank web tuple containing C before the web tuple containing B.

4.2 Data Visualization Operators

Presently, in the WICM we have a set of web operators to manipulate information from the WWW. Web information is materialized and displayed in the form of web tuples stored in a web table. This approach of displaying information to the user has few shortcomings:

- It does not provide us with the ability to see the overall structure of the information captured in the web table. It is not possible for a user to visualize how one web tuple in a web table is related to another.
- The set of web tuples in a web table may contain duplicate web documents. There is no mechanism to provide a *coalesced view* of the set of web tuples. A coalesced view allows a user to browse lesser number of directed connected graphs when locating information.
- It does not allow a user to group web tuples based on *related information content*, or *similar (or identical)* web sites. A user has to manually probe each web tuple to find these information. However, these information cannot be grouped together in our web table.
- The set of web tuples are materialize in web table. There is no other representation of these web tables. For example, the collective view of these web tuples can be stored as a set of directed graphs having lesser number of nodes or links as compared to the original web table.

To resolve the above difficulties, we have introduced to following data visualization operators:

- `Web Nest:` This operator allows one to visualize relationships between web tuples in a web table. It provides an overview of the information space harnessed from the WWW and shows how these information are topically related.
- `Web Unnest:` This operator returns the original web table from a set of directed connected graphs created by the web nest operation.
- `Web Coalesce:` This operator coalesces duplicate nodes to reduce the size of the number of directed connected graphs in a web table. A coalesced web table is a set of condensed graphs and allow a user to browse lesser number of graphs (web tuples).

- `Web Expand:` This operator expands a coalesced web table to recover the original set of web tuples.
- `Web Pack:` This operator groups a web tuples based on related (or similar) information content, and similar (or identical) web sites.
- `Web Unpack:` This operator returns the original set of web tuples from the packed web table.
- `Web Sort:` This operator sorts web tuples in *ascending* or *descending* order based on the total number of nodes or link types (local, global or interior) in a web tuple.

These web operators take as input a set of web tuples of a web table and provide a different view of the tuples as output. This gives users the flexibility to view documents in perspectives that are more meaningful. These operators provide different storage representation of web tuples which will help in optimizing the query, storage and maintenance cost. These different representations may also lead to inconsistency problems with respect to the original web table. Currently, we are investigating these problems.

5 Web Data Mining

The resulting growth in on-line information combined with the almost unstructuredness of web data necessitates the development of powerful yet computationally efficient web mining tools. Web mining can be defined as the discovery and analysis of useful information from WWW data. Web mining involves three types of data; data on the WWW, web documents structure and the data on users who browse web pages. Thus, web data mining should focus on three issues; content-based mining [17], web usage mining [23] and web structure mining. Web content mining describes the automatic search of information resources available on-line. Web usage mining includes the mining of data from the server access logs, user registration or profiles, user sessions or transactions, etc. Web structure mining involves mining web document's structures and links. A survey of some of the emerging tools and techniques for web usage mining is given in [21]. One of the important areas in WHOWEDA involves the development of tools and techniques for mining useful information from the web.

The web contains a mix of many different data types, and in a sense subsumes text data mining, database data mining, image mining, and so on. The web contains additional data types that are not available in a large scale before, including hyperlinks and massive amounts of (indirect) user usage information. Spanning across all these data types is the dimension of time, since data on the web changes over time. Finally, there is data that are generated dynamically, in response to user input and programmatic scripts. In WHOWEDA, we primarily focus on mining useful information from these different data types. For further information related to web mining in WHOWEDA refer to [7].

5.1 Web Bags and Knowledge Discovery

Most of the search engines fail to handle the following knowledge discovery goals:

- From query results returned by search engines, a user may wish to locate the most *visible* Web sites [4] or documents for reference. That is, sites or documents which can be reached by many paths (high fan-in).
- Reversing the concept of visibility, a user may wish to locate the most *luminous* Web sites [4] or documents for reference. That is, web sites or documents which have the most number of outgoing links.
- Furthermore, a user may wish to find out the most traversed path for a particular query result. This is important since it helps the user to identify the set of most popular interlinked Web documents which are traversed frequently to obtain the query result.

We have introduced the concept of web bag in [5] and used web bags for knowledge discovery. Informally, a web bag is a web table containing multiple occurrences of the identical web tuples. A web tuple is a set of inter-linked documents retrieved from the WWW that satisfy a query graph. A web bag may only be created by projecting some of the nodes from the web tuples of a web table using the web project operator. A web project operator is used to isolate data of interest, allowing subsequent queries to run over a smaller, perhaps more structured web data. Unlike its relational counterpart, a web project operator does not eliminate identical web tuples autonomously. The projected web table may contain identical web tuples, thus forming a web bag. The duplicate elimination process is initiated explicitly by a user. Autonomous duplicate elimination may hinder the possibility of discovering useful knowledge from a web table. This is due to the fact that these knowledge may only be discovered from the web bags.

5.2 Warehouse Concept Mart (WCMart)

Due to the large amount of data on the WWW, knowledge discovery from web data becomes more and more complex. We propose the building of concept hierarchies from web documents and use them in discovering new knowledge. We call the collection of concepts a Warehouse Concept Mart (WCMart). Concept marts are built by extracting and generalizing terms from web documents to represent classification knowledge of a given class hierarchy. For unclassified words, they can be clustered based on their common properties. Once the clusters are decided, the keywords can be labeled with their corresponding clusters, and common features of the terms are summarized to form the concept description. We may associate a weight at each level of concept marts to evaluate the importance of a term with respect to the concept level in the concept hierarchy. Concept marts can be used for the following:

- *Intelligent answering of web queries:* Knowledge discovery using concept marts facilitates querying web data and intelligent query answering in web warehousing system. A user can supply the threshold for a given key word

in the concept mart and the words with the threshold above the given value can be taken into account while answering the query. The query can also be answered using different levels of concept marts [14] or can provide approximate answers [19].

- *Ranking result tuples of a query:* In our model, tuples returned as a result of a web query are stored in a web table. We use warehouse concept marts for ranking these tuples so that the most relevant tuples are returned in response to a user's query. The user may rank tuples interactively by specifying various threshold values and concept levels.
- *Global information coupling across the WWW:* In our model, we introduce the concept of web information coupling. It refers to an association of related web documents. We use concept marts to define this "association" among web documents. Since the coupling is initiated by the user, he may supply threshold values for keywords in the concept mart to be used in the coupling.
- *Web mining:* Concept marts can also be used for web data mining. Web mining so far in the literature is restricted to mining web access patterns and trends by examining the web server log files [23]. An alternative is to make use of web concept marts in generating some useful knowledge. We may use association rule techniques to mine associations between words appearing in the concept mart at various levels, and the query graph. Mining knowledge at multiple levels may help WWW users discover interesting rules which are difficult to find otherwise.
- *Characterization of web tables:* In a web warehouse, we store tuples in web tables. We need to categorize these tables so that a user's query can be directed to the appropriate table(s). Based on the classification of web tables, a user may also specify tables to be used for evaluating his query. We categorize the web tables using the concept marts. We generate concept marts from web tables and classify a set of tables by known classes or concepts.

6 Conclusions

In this paper, we have reported an overview of a Web Warehousing Project (WHOWEDA) [22] at the Nanyang Technological University, Singapore and discussed some interesting research ideas in that context. Building a warehouse that accommodates data from the WWW has required us to rethink nearly every aspect of conventional data warehousing and relational technology. This paper brings together some of the important techniques required for designing a web warehouse and generating useful knowledge. In particular, our focus is on web data model and an algebra for web information access, manipulation and visualization. We have also discussed the motivation for developing a Web Concept Mart and its application in web warehousing. Furthermore, we have outline the problem of web mining and maintenance of web information. We believe that issues presented here will serve as an interesting example for further discussion.

References

1. http://www.cais.ntu.edu.sg:8000/~ whoweda/.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Weiner. The Lorel Query Language for Semistructured Data. *Journal of Digital Libraries*, 1(1):68-88, April 1997. 95
3. G. Arocena, A. Mendelzon. WebOQL: Restructuring Documents, Databases and Webs. *Proceedings of International Conference on Data Engineering* , Orlando, Florida, February 1998.
4. T. Bray. Measuring the Web. *Proceedings of the 5th International World Wide Web Conference (WWW)*, Paris, France, 1996. 102
5. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Web Bags: Are They Useful in A Web Warehouse? *Proceedings of 5th International Conference of Foundation of Data Organization (FODO'98)*, Kobe, Japan, November 1998. 96, 97, 102
6. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Data Visualization in a Web Warehouse. *Proceedings of International Workshop on Data Warehousing and Data Mining (DWDM'98) (in conjunction with ER'98)*, Singapore, 1998.
7. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Web Mining in Whoweda: Some Issues.*PRICAI'98 Workshop on Knowledge Discovery and Data Mining*, Singapore, 1998. 101
8. S. Bhowmick, S. K. Madria, W.-K. Ng, E.-P. Lim. Bags in A Web Warehouse: Design and Analysis. *Submitted for publication.*
9. S. Bhowmick, W.-K. Ng, E.-P. Lim. Information Coupling in Web Databases. *Proceedings of the 17th International Conference on Conceptual Modelling (ER'98)*, Singapore, 1998. 94, 95, 96, 97
10. S. S. Bhowmick, W.-K. Ng, E.-P. Lim, S. K. Madria. Join Processing in Web Databases. *Proceedings of the 9th International Conference on Database and Expert Systems Application (DEXA)*, Vienna, Austria, 1998. 99
11. P. Buneman, S. Davidson, G. Hillebrand, D. Suciu. A query language and optimization techniques for unstructured data. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Canada, June 1996. 95
12. M. Fernandez, D. Florescu, A. Levy, D. Suciu. A Query Language for a Web-Site Management Systems *SIGMOD Record*, 26(3), Sept, 1997.
13. T. Fiebig, J. Weiss, G. Moerkotte. RAW: A Relational Algebra for the Web. *Workshop on Management of Semistructured Data (PODS/SIGMOD'97)*, Tucson, Arizona, May 16, 1997. 94
14. J. Han, Y. Huang, N. Cercone, Y. Fu. Intelligent Query Answering by Knowledge Discovery Techniques. *IEEE Transactions of Knowledge and Data Engineering.*, 8(3):373 – 390, 1996. 103
15. D. Konopnicki, O. Shmueli. W3QS: A Query System for the World Wide Web. *Proceedings of the 21st International Conference on Very Large Data Bases*, Zurich, Switzerland, 1995. 94
16. L.V.S. Lakshmanan, F. Sadri., I.N. Subramanian. A Declarative Language for Querying and Restructuring the Web *Proceedings of the Sixth International Workshop on Research Issues in Data Engineering*, February, 1996. 94, 95
17. S. H. Lin, C. S. Shih, M. C. Chang Chen et al. Extracting Classification Knowledge of Internet Documents with Mining Term Associations: A Semantic Approach. *Proceedings of the Sixth International Workshop on Research Issues in Data Engineering*, February, 1996. 101

18. M. Liu, T. Guan, L. V. Saxton. Structured-Based Queries over the World Wide Web. *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, Singapore, 1998.
19. S. K. Madria, M. Mohania, J. F. Roddick. A Query Processing Model for Mobile Computing using Concept Hierarchies and Summary Databases. *Submitted for publication*. 103
20. A. O. Mendelzon, G. A. Mihaila, T. Milo. Querying the World Wide Web. *Proceedings of the International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami, Florida. 94, 95
21. B. Mobasher, R. Cooley, J. Shrivastava. Web Mining: Information and Pattern Discovery on the World Wide Web. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997. 101
22. W. K. Ng, E.-P. Lim, C. T. Huang, S. Bhowmick, F. Q. Qin. Web Warehousing: An Algebra for Web Information. *Proceedings of IEEE International Conference on Advances in Digital Libraries (ADL'98)*, Santa Barbara, California, April 22–24, 1998. 94, 95, 96, 103
23. O. R. Zaine, M. Xin, J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. *Proceedings of IEEE International Conference on Advances in Digital Libraries (ADL'98)*, Santa Barbara, California, April 22–24, 1998. 101, 103

Extending the E/R Model for the Multidimensional Paradigm

Carsten Sapia, Markus Blaschka, Gabriele Höfling, and Barbara Dinter

FORWISS (Bavarian Research Center for Knowledge-Based Systems)
Orleansstr. 34, D-81667 Munich, Germany
Email: {sapia, blaschka, hoefling, dinter}@forwiss.tu-muenchen.de

Abstract. Multidimensional data modeling plays a key role in the design of a data warehouse. We argue that the Entity Relationship Model is not suited for multidimensional conceptual modeling because the semantics of the main characteristics of the paradigm cannot be adequately represented. Consequently, we present a specialization of the E/R model - called Multidimensional Entity Relationship (ME/R) Model. In order to express the multidimensional structure of the data we define two specialized relationship sets and a specialized entity set. The resulting ME/R model allows the adequate conceptual representation of the multidimensional data view inherent to OLAP, namely the separation of qualifying and quantifying data and the complex structure of dimensions. We demonstrate the usability of the ME/R model by an example taken from an actual project dealing with the analysis of vehicle repairs.

1 Introduction

Multidimensional data modeling plays a key role during the design of a data warehouse. The multidimensional warehouse schema offers an integrated view on the operational data sources. Consequently, it serves as the core of the data warehouse and as the basis for the whole warehouse development and maintenance cycle. Due to this central role sufficient attention should be paid to the development of this schema. Figure 1 sketches the process of the schema design in data warehousing environments. The schema is mainly influenced by user requirements and the availability and structure of the data in operational systems. Most warehousing projects take an evolutionary approach[1], i.e. start with a prototype providing a certain functionality and set of data. This prototype will be further adopted according to the changing and growing requirements gained from users' feedback. Thus the user requirements are subject to frequent changes making schema evolution an important issue. To assure the flexibility and re-usability of the schema in such an environment, the model must be specified on a conceptual level (e.g. using the Entity Relationship Model). This means especially that it must not assume any facts that are the result of further design steps

[1] both in our experience from industrial projects [9] and in the warehouse literature, see e.g. [11]

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 105 -116, 1999.

e.g. the decision which database technology is to be used (multidimensional vs. relational).

For OLAP and data warehouse systems this is even more important as the most common design methodologies mix up the conceptual and the logical/physical design. Currently the state of art in dimensional modeling is the use of implementation (mostly even tool specific) formalisms for data modeling. For example, the ubiquitous star schema is not conceptual in the sense that it assumes the relational implementation and contains further decisions (e.g. denormalization) that should be subject of the physical design phase.

There is a consensus ([11], [13], [15]) that the multidimensional paradigm comes very close to the inherent structure of the problem domain (decision support systems). In this paper we investigate the special requirements of the multidimensional paradigm. We argue that the established conceptual design methods used for relational (e.g. the Entity Relationship Model [4]) or object-oriented systems do not offer the necessary support to reflect the multidimensional data model in a natural and intuitive way. Moreover, some of the multidimensional semantics is lost when expressing a multidimensional schema with these techniques. This means that the semantics must be represented informally which makes them unusable for the purpose of automatic generation (e.g. automatic generation of database schemes or query tools).

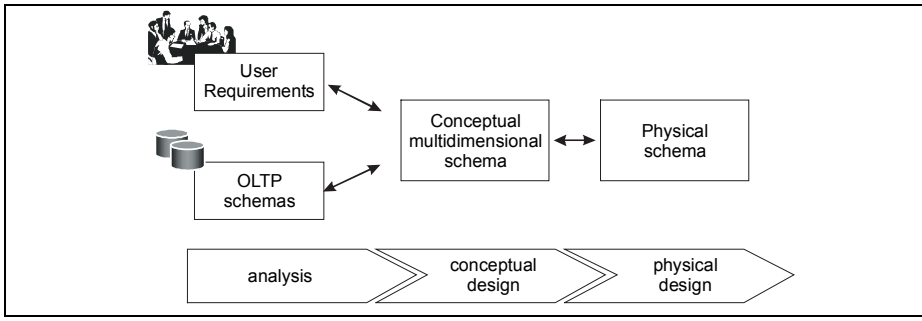


Fig. 1. Schema design process in data warehousing environments

Possible approaches to an expressive conceptual multidimensional model are to build a new model from scratch (which also means additional effort for its formal foundation) or to use an existing, general-purpose model and modify it so that the special characteristics of the multidimensional paradigm can be expressed.

Consequently, this paper presents a multidimensional specialization of the E/R model - called Multidimensional E/R Model (ME/R Model). By basing our approach on an established model we enable the transfer of the research results published in the context of the E/R model. This includes especially the work about automatic schema generation and formal foundation of the semantics. Furthermore, it is possible to make use of the proven flexibility of the well accepted E/R model.

The remainder of this paper is structured as follows: section 2 informally introduces the multidimensional paradigm and states the special requirements of OLAP applications regarding the data model. Section 3 describes the specializations of the E/R model that are necessary to fulfil these requirements and defines the ME/R model. In section 4 we investigate the expressive power of the ME/R model. To dem-

onstrate the feasibility of our approach we model a real world example (section 5). Finally, we present related (section 6) and future work (section 7).

2 The Multidimensional Paradigm

The multidimensional paradigm is useful for a multitude of application areas (e.g. GIS, PACS, statistical databases and decision support). For the purpose of this paper we focus on typical OLAP applications. For example a vehicle manufacturer might want to analyze the vehicle repairs to improve his product, define new warranty policies and to get information about the quality of the garages.

Often a cube metaphor ([3]) is used to represent this data view as shown in figure 2. Such a cube corresponds to a subject of analysis called *fact* (e.g. vehicle repair). The cells of the data cube contain the (mostly numerical) *measures* (also called *quantifying data*) describing the fact (e.g. costs and duration of the vehicle repair). The axes of the cube (called *dimensions* or *qualifying data*) represent different ways of analyzing the data (e.g. vehicle and time of the repair).

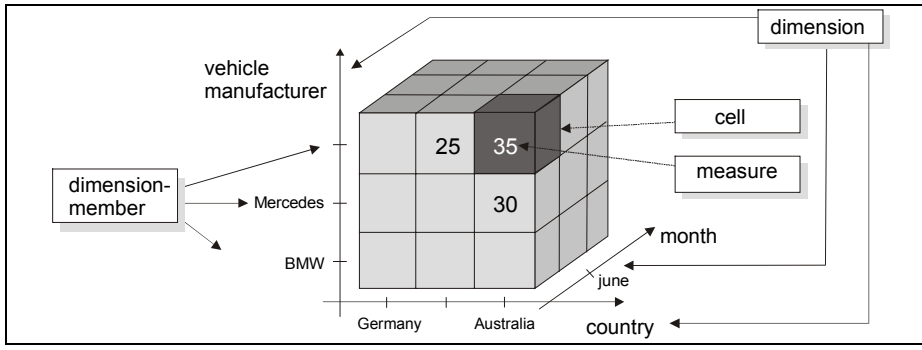


Fig. 2. A visualization of a multidimensional schema using the cube metaphor

This data view is similar to the notion of arrays. However, with arrays the dimensions of the multidimensional data space are only structured by a linear order defined on the indexes. For OLAP applications this is not sufficient because from the point of view of the OLAP end-user, the elements (respectively instances) of an OLAP dimension (called dimension members) are normally not linearly ordered (e.g. garages)[2].

Instead classification hierarchies containing levels are used for the structuring of dimensions. A hierarchy level contains a distinct set of members. Different levels correspond to different data granularities (e.g. daily figures vs. monthly figures) and ways of classification (e.g. geographic classification of garages vs. classification of garages by type). Level A is classified according to level B if a classification of the elements of A according to the elements of B is semantically meaningful to the application (e.g. the level 'days' are classified according to 'month').

[2] A prominent exception to this rule is the time dimension that possesses an inherent order

A level can be classified according to any number of levels thus forming multiple hierarchies on a single dimension. For example, garages can be classified by their geographical location and their type (see example in section 5). Another special case of hierarchies are alternative paths. This type of hierarchy occurs if several classification paths exist between two levels. An example for this is the classification of cities by geographical regions and federal districts (see figure 3).

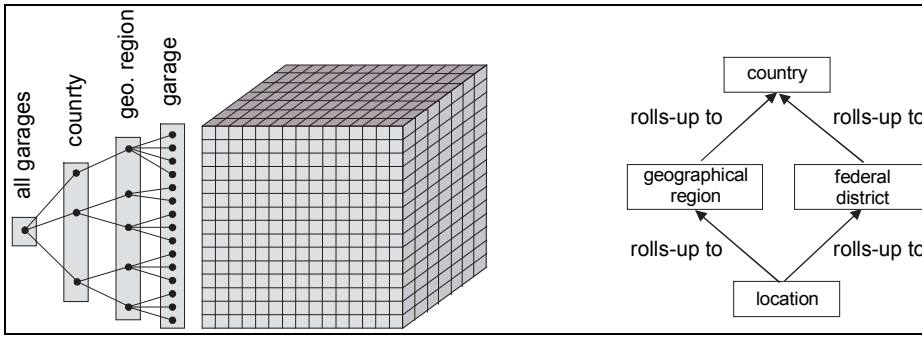


Fig. 3. Hierarchy levels structure the dimensions(left). Alternative pathes within a dimension(right)

Another orthogonal way of structuring dimensions from a users point of view is the use of dimension level attributes. These attributes describe dimension level members but do not define hierarchies (e.g. the name and address of a customer or the name of the region manager).

Not only qualifying data but also quantifying data possesses an inherent structure. In most applications different measures describing a fact are common (e.g. for a vehicle repair it might be useful to measure the duration of the repair, the costs for parts being exchanged and the cost for the wages). That means that a cell of the cube does contain more than one numeric value. Some of these measures are derived, i.e. they can be computed from other measures and dimension attributes (e.g. total repair cost is the sum of part costs and costs for wages).

A complex schema can contain more than one cube. This becomes necessary if an application requires the analysis of different facts (e.g. vehicle sales and repairs) or if not all of the measures are dependent on the same set of dimensions.

These multiple cubes can share dimensions (e.g. the time dimension). This does not necessarily mean that these cubes measure data using the same granularity. For example vehicle sales might be recorded and analyzed on a weekly basis, while the vehicle repairs are recorded daily (see section 5).

Regarding the multidimensional paradigm it is obvious, that the E/R model is not very well suited for the natural representation of multidimensional schemas. The inherent separation of qualifying and quantifying data cannot be expressed as all entity sets are treated equally by the E/R model. Furthermore the semantics of the complex structure of the dimensions (classification relationship between dimension levels) is an integral part of the multidimensional paradigm that is too specific to be modeled as a general purpose relationship.

3 The Multidimensional E/R Model

In order to allow the natural representation of the multidimensional semantics inherent to OLAP schemas, the E/R model is specialized. Of course, there are several possible ways to achieve this goal. Our design was driven by the following key considerations:

- *Specialization of the E/R model*: All elements that are introduced should be special cases of native E/R constructs. Thus, the flexibility and expressiveness of the E/R model is not reduced.
- *Minimal extension of the E/R model*: The specialized model should be easy to learn and use for an experienced E/R modeler. Thus, the number of additional elements needed should be as small as possible. A minimal set of extensions ensures the easy transferability of scientific results (e.g. formal foundations) from the E/R model to the ME/R model by discussing only the specific extensions.
- *Representation of the multidimensional semantics*: Despite the minimality, the specialization should be powerful enough to express the basic multidimensional semantics, namely the separation of qualifying and quantifying data and the hierarchical structure of the qualifying data.

A lot of variations of the E/R model (for an overview see e.g. [18]) have been published since the first proposal of Chen. For the purpose of this paper we use a very basic version of the E/R model. We formally describe our specialized E/R model using the meta modeling approach. We adhere to the four layer technique of the ISO/IRDS standard for metadata [12]. Figure 4 shows the meta model of our M/ER model (Dictionary Definition Layer of the IRDS). The part with the white background shows the meta model of the E/R model we use as a foundation. For the purpose of describing the meta model, we make use of an extended version of the E/R model which allows the concept of generalization. This is done to increase the readability of the meta model. However, the decision which type of constructs are allowed in the E/R model itself (and thus the ME/R model) is left open to the modeler.
Following our key considerations we introduce the following specialization:

- a special entity set: dimension level,
- two special relationship sets connecting dimension levels:
  - a special n-ary relationship set: the '*fact*' relationship set and
  - a special binary relationship set: the '*classification*' relationship set.

Since the semantic concept 'dimension level' is of central importance, we introduce a special entity set for dimension levels.

To model the structure of qualifying data we introduce a special binary relationship set: the classification relationship. It relates a dimension level A to a dimension level B representing concepts of a higher level of abstraction (e.g. city *is classified* according to country). The classification graph is defined as follows: $RG = (E,V)$ with E being the finite set of all dimension levels $e_1,...,e_k$ and $V = \{ (e_i,e_j) \mid i \neq j \wedge 1 \leq i,j \leq k \wedge e_i$ *is classified according to* $e_j \}$. Due to the special semantics of the classification relation, no cycles must be contained in the graph as this could lead to semantically not reasonable infinte roll-up pathes (e.g. day is classified according to month and month

is classified according to day). This means the following global integrity constraint must be fulfilled (→* denotes the transitive closure of the *classification relation*):

$$\forall\ e_i, e_j \in E : e_i \rightarrow^* e_j \Rightarrow i \neq j$$

Thus the classification graph *RG* is a directed acyclic graph (DAG). The name attribute of the classifiaction relation set describes the criteria of classification. (e.g. 'lives in' for the classification relationship set connecting 'customer' and 'geographical region')

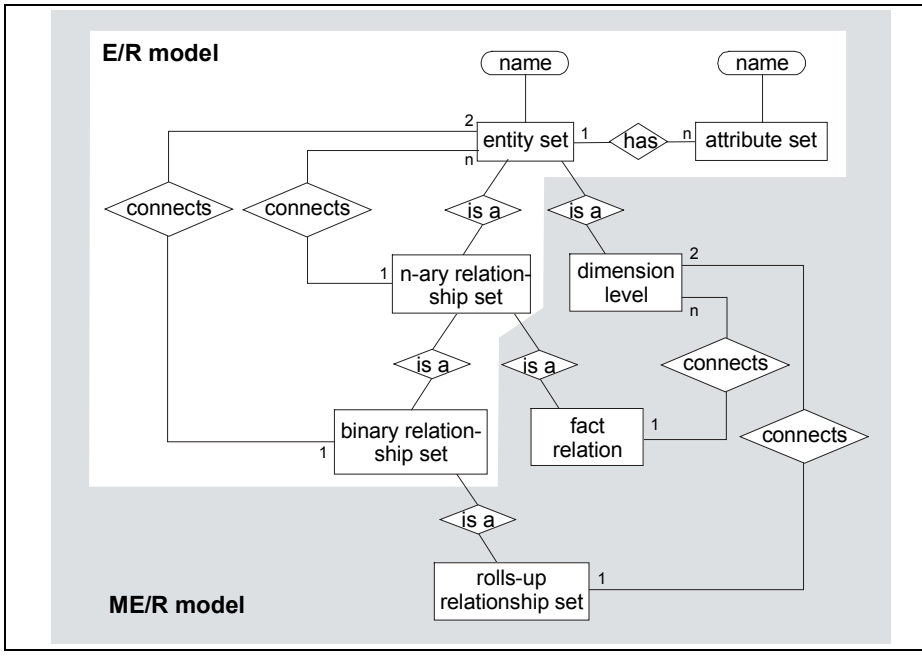


Fig. 4. The meta model of the ME/R model is an extension of the meta model of E/R.

The fact relationship set is a specialization of a general n-ary relationship set. It connects n different dimension level entities. Such a relation represents a fact (e.g. vehicle repair) of dimensionality n. A description of the fact is used as the name for the set. The directly connected dimension levels are called *atomic* dimension levels.

The fact relationship set models the inherent separation of qualifying and quantifying data. The attributes of the fact relationship set model the measures of the fact (quantifying data) while dimension levels model the qualifying data.

To distinguish our specialized elements from the native E/R modeling elements and to enhance the understandability of the graphical model, we use a special graphical notation for dimension level sets, fact relationship sets, and classification relationship sets (figure 5).

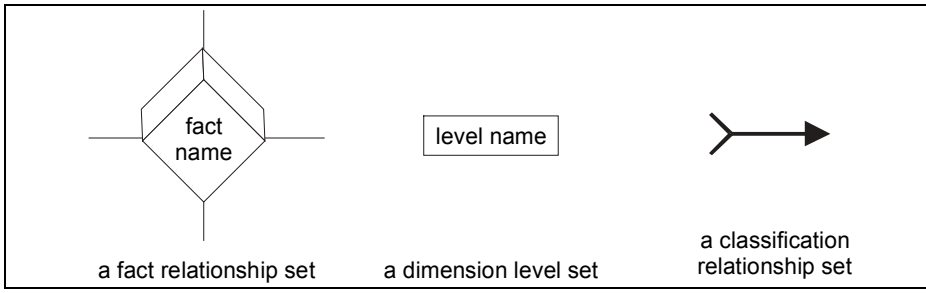


Fig. 5. The graphical notation of the ME/R elements.

4 Distinctive Features of the ME/R Model

After having introduced the ME/R model, we now investigate how the ME/R model fulfills the requirements of the multidimensional paradigm. An example for modeling a real-world scenario can be found in the next section.

A central element in the multidimensional paradigm is the notion of dimensions that span the multidimensional space. The ME/R model does not contain an explicit counterpart for this concept. This is not necessary because a dimension consists of a set of dimension levels. The information which dimension-levels belong to a given dimension is included implicitly within the structure of the classification graph. Formally, the fact relationship identifies the n atomic dimension levels $e_{i1},..e_{in}$. The according dimensions D_k are the set of the dimension levels that are included in the subgraph of the classification graph *RG(E,V)* defined by the atomic level.

$$D_k = \left\{ e \in E \mid e_{i_k} \rightarrow^* e \right\} \quad 1 \le k \le n$$

The hierarchical classification structure of the dimensions is expressed by dimension level entity sets and the classification relationships. As previously noted, the classification relationship sets define a directed acyclic graph on the dimension levels. This enables the easy modeling of multiple hierarchies, alternative paths and shared hierarchy levels for different dimensions (e.g. customer and garage in figure 7). Thus no redundant modeling of the shared levels is necessary. Dimension level attributes are modeled as attributes of dimension level entity sets. This allows a different attribute structure for each dimension level.

By modeling the multidimensional cube as a relationship set it is possible to include an arbitrary number of facts in the schema thus representing a 'multi-cube model'. These different cubes and their shared dimensions can be expressed as shown in figure 7. Notably the schema also contains information about the granularity level on which the dimensions are shared. This information is for example necessary for the design of multidimensional joins during further development steps.

Regarding measures and their structure the ME/R model allows record structured measures as multiple attributes are possible for one fact relationship set. The semantic information that some of the measures are derived cannot be included in the model. Like the E/R model the ME/R model captures the static structure of the application

domain. The calculation of measures is a functional information and should not be included in the static model. An orthogonal functional model should capture these dependencies.

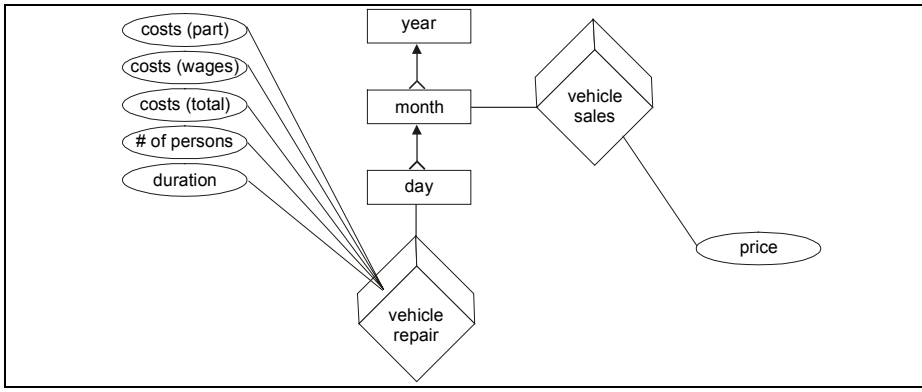


Fig. 6. Multiple cubes sharing a dimension on different levels

5 Applying the ME/R Model (Case Study)

To demonstrate the feasibility of our ME/R model, we present a real application. The following example is taken from a project with an industrial partner [10]. An automobile manufacturer stores data about repairs of vehicles. Among other, the date of repair, properties of the vehicle (e.g. model), information about the specific repair case (e.g. costs, number of garage employees involved, duration of the repair), data about the garage doing the repair, and data about the customer who owns the vehicle are stored.

Typical examples queries for this scenario are: "Give me the average total repair costs per month for garages in Bavaria by type of garage during the year 1997" or "Give me the five vehicle types that had the lowest average part costs in the year 1997"

The first design step is to determine which data are dimensions and which are facts. We assume that the repair costs (broken down by part costs, wages and total) for a specific vehicle (owned by a customer) for a specific garage are given on a daily basis. Then the facts (quantifying data) are the repair costs (parts, wages, total). Vehicle, customer, garage and day are the corresponding dimensions (qualifying data) and because they are at the finest granularity also the atomic dimension levels. Thus, the *fact* relationship connects the *vehicle repair fact* with the dimensions vehicle, customer, garage and day. The *classification relation*ships are shown in figure 7 which contains the complete ME/R diagram for this case study. The *fact* relationship in the middle of the ME/R diagram connects the atomic dimension levels. Each dimension is represented by a subgraph that starts at the corresponding atomic level (e.g. the time dimension starts at the dimension level day and comprehends also month and year). The actual facts (part costs, wages, total costs, number of persons involved and duration of the repair) are modeled as attributes of the *fact* relationship. The dimension

hierarchies are depicted by the *classification* relationships (e.g. vehicle *is classified* according to model and brand). Additional attributes of a dimension level (e.g. age or income of a customer) are depicted as dimension attributes of the corresponding dimension level.

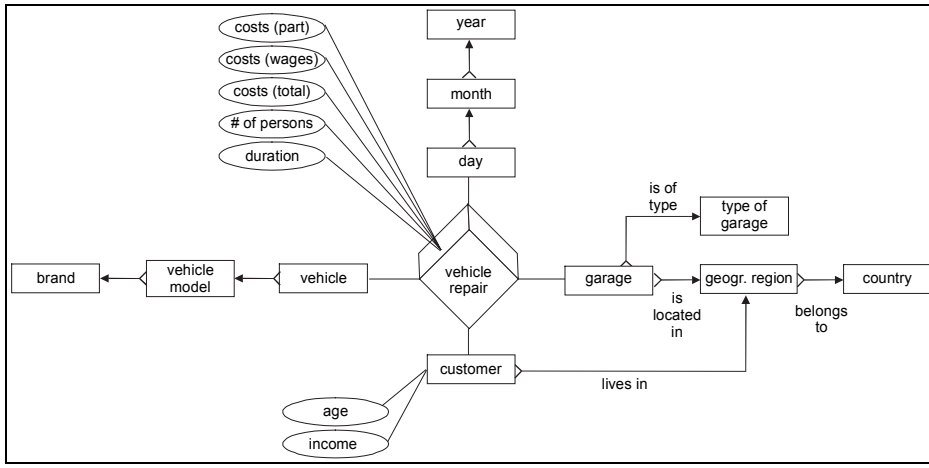


Fig. 7. The ME/R diagram for the analysis of vehicle repairs

Notably, the schema contains a *classification relation*ship between the entities *'customer'* and *'geographic region'* and between *'garage'* and *'geographic region'*. This shows a distinctive feature of our model: levels of different dimensions may by classified according to a common parent level. This might imply that the dimensionality of the cube is reduced by one when executing a roll-up operation along this classification. However, this is not the case as the model only captures the semantical fact, that the same type of classification (geographical classification) is used in both dimensions. During later phases of the development cycle this information can be used to avoid redundancies as the geographical regions only have to be stored once. The corresponding data cube however still contains two dimensions that contain the same members (customer geographical region and garage geographical region).

Since our ME/R model is a **specialization** of the E/R model, regular E/R constructs can also be used in ME/R diagrams. In our example, the entity *vehicle* can be extended e.g. to distinguish between cars and trucks. This scenario is shown in figure 8. We use the *isa* relationship to model the categorization of vehicles. The extended diagram (i.e. with 'regular' E/R constructs and special ME/R constructs) further allows us to model additional features of the subtypes of our entity *vehicle*. Features [14] are attributes that are only meaningful in a subclass but not in a superclass. Different subclasses may have different features. For example, for a *vehicle* in general one might store attributes like *length*, *width, height, colour,* or *horse power*, but a feature like *loading capacity* or *loading area* in m² is only meaningful for trucks. For a car on the other hand, it might be useful to store the *number of seats* or the *type of gear* (i.e. manual or automatic). Thus, using these combined E/R and ME/R modeling technique, features as introduced in [14] can be modeled on a conceptual level.

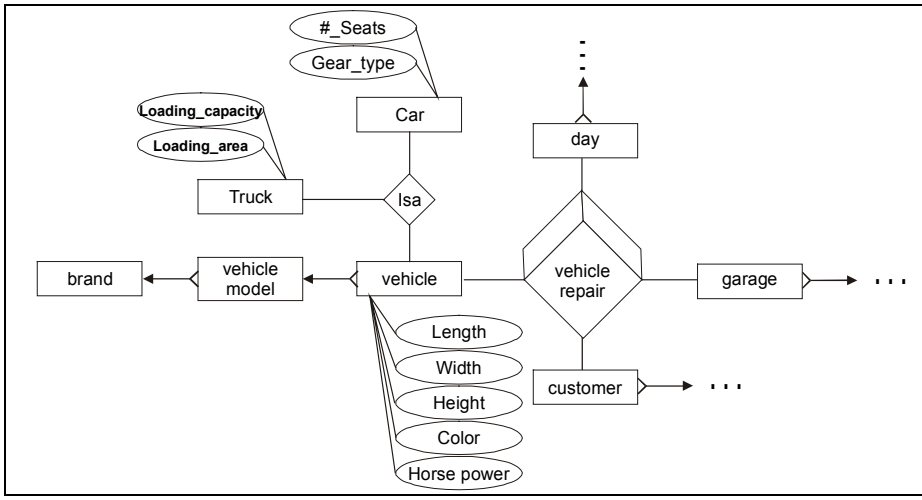


Fig. 8. Combining ME/R notation with classical E/R constructs

6 Related Work

A lot of publications are available concerning 'multidimensional data modeling'. Unfortunately only very few recognize the importance of the separation of conceptual and logical/physical issues. This is largely because the development in this area has so far been driven by the product vendors of OLAP systems. To our knowledge only very few papers investigating a graphical conceptual (i.e. implementation independent) data modeling methodology for multidimensional database have been published. Ralph Kimball proposes the design of Data Warehouses using a multidimensional view of the enterprise data. He presented a 'multidimensional modeling manifesto' [13]. However, his approach is not conceptual in the sense that it is not independent of the implementation (a relational implementation in the form of a 'star schema' is assumed).

In the area of statistical databases, graphical conceptual models to capture the structure and semantics of statistical tables have been proposed (e.g. [16], [17]) for a long time. The data warehouse research community focused mainly on physical issues (e.g. [9]) of data warehouse design. Quite a lot of work has also been done to formalize or extend ([7],[14]) the multidimensional data model (see [2] for a comparison) and to define query languages for these data models ([1]). However, these formalisms are not suited for conceptual modeling of user requirements. Our work supplements these papers by providing a graphical conceptual layer (as the E/R model provides for the relational paradigm).

Nevertheless, recently the deficit in conceptual models has been recognized. [6] proposes a formal logical model for OLAP systems and showed how it can be used in the design process. The paper suggests a bottom-up approach to data warehouse design. The authors assume an integrated E/R schema of the operational data sources and give a methodology to transform this schema into a dimensional graph which can be translated into the formal MD model. Our model is more suited to a top-down

approach modeling the user requirements independently from the structure of the operational systems.

[8] also proposes a conceptual model called dimensional fact (DF) scheme. They give a graphical notation and a methodology to derive a DF model form the E/R models of the data sources. Although the technique supports semantically rich concepts it is not based on a formal data model. Our approach is to specialize a well researched and formally founded model. Furthermore, the notation does not allow the modeling of alternative paths which we believe is an important requirement.

In [14] Lehner et al. present a conceptual multidimensional data model. They argue that the common classification hierarchies are not sufficient for all types of applications. Therefore, they propose feature descriptions as a complementary mechanism for structuring qualifying information. As the main focus of the paper is the extension of the paradigm, no graphical notation (apart from the cube visualization) is provided.

7 Conclusions and Future Work

We started from the fact that the multidimensional paradigm plays a central role in the data warehouse and OLAP design process. However the fundamental semantics of this paradigm cannot be adequately expressed using the E/R model. Consequently, we proposed ME/R, a specialization of the E/R model especially suited for the modeling of OLAP applications. We also defined a graphical notation for the new elements which allows intuitive graphical diagrams. Our technique supports the easy modeling of multidimensional semantics (namely the separation of qualifying and quantifying data and the complex structure of dimensions). Multiple hierarchies, alternative paths and shared dimension levels can be naturally expressed. By designing ME/R as a specialization of the common E/R model we ensure a shallow learning curve and a high intuitivity of the diagrams. Since the modeler can combine ME/R elements with classical E/R elements semantically rich models can be built. Finally, we demonstrated the flexibility and usefulness of our approach by modeling a real world example.

The ME/R model can serve as the core of a full scale data warehouse design methodology. Using the ME/R model it is possible to capture the multidimensional application semantics during the conceptual design phase of a data warehouse. This information can be used during later phases (physical design and implementation) of the data warehouse process for (semi-) automatic system generation. A first step in this direction will be the mapping of the ME/R model to the formal logical multidimensional data models that were proposed recently.

The ME/R model allows to capture the static data structure. The modeling of dynamical (e.g. anticipated query behavior) and functional (e.g. the additivity of measures along dimensions or the functional relationship between hierarchy levels) aspects deserve a deeper study. Currently we are investigating a dynamic and a functional model supplementing the static model (analogous to the OMT) and study the interrelationship between those models. Additionally, we are working on a classification of multidimensional schema evolution operations (e.g. ‘add dimension level’) and examine the impacts of these operations. To this end, we evaluate schema evolution approaches from object-oriented databases and investigate their feasibility in the multidimensional case.

References

[1] A, Bauer, W. Lehner: *The Cube-Query-Language (CQL) for Multidimensional Statistical and Scientific Database Systems,* Proc. of the 5th CIKM, Melbourne 1997.
[2] M. Blaschka, C. Sapia, G. Höfling, B. Dinter: Finding Your Way through Multidimensional Data Models, DWDOT Workshop (DEXA 98), Vienna
[3] S. Chaudhuri, U. Dayal: *An Overview of Data Warehousing and OLAP Technology.* SIGMOD Records 26(1), 1997
[4] P.P-S. Chen: The Entity Relationship Model – Towards a Unified View of Data. ACM TODS Vol. 1, No. 1, 1976
[5] E. F. Codd: Extending the Database Relational Model to Capture More meaning. ACM TODS Vol. 4, No. 4 (December 1979)
[6] L. Cabibbo, R. Torlone: *A Logical Approach to Multidimensional Databases.* EDBT 1998.
[7] S. Dekeyser, B. Kuijpers, J. Paredaens, J. Wijsen: *The nested datacube model for OLAP* in Advances in Database Technology, LNCS, Springer Verlag
[8] M. Golfarelli, D. Maio, S. Rizzi, *Conceptual design of data warehouses from E/R schemes*, Proc. 31st Hawaii Intl. Conf. on System Sciences, 1998.
[9] V. Harinarayan, A. Rajaraman, J. D. Ullman: *Implementing Data Cubes Efficiently.* Proc. SIGMOD Conference, Montreal, Canada, 1996
[10] G. Höfling, M. Blaschka, B. Dinter, P. Spiegel, T. Ringel: *Data Warehouse Technology for the Management of Diagnosis Data* (in German), in Dittrich, Geppert (eds.): *Datenbanksysteme in Büro, Technik und Wissenschaft* (BTW), Springer, 1997.
[11] W. H. Inmon: *Building the Data Warehouse*, 2nd edition, John Wiley &Sons, 1996
[12] IRDS Framework ISO/IEC IS 10027, 1990
[13] R. Kimball: *A Dimensional Modeling Manifesto,* DBMS Magazine, August 1997,
[14] W. Lehner, T. Ruf, M. Teschke: CROSS-DB: *A Feature-Extended Multidimensional Data Model for Statistical and Scientific Databases*, Proc. of the CIKM'96, Maryland.
[15] Micro Strategy Inc.: The Case For Relational OLAP, White Paper. 1995,
[16] M. Rafanelli, A. Shoshani: *STORM : A Statistical Object Representation,* SSDBM 90
[17] S.Y.W. Su : SAM*: *A Semantic Association Model for Corporate and Scientific-Statistical Databases,* in: Journal of Information Sciences 29, 1983
[18] T.J. Teorey: *Database Modeling and Design*, 2nd edition, Morgan Kaufmann 1994

Numerical Aspects in the Data Model of Conceptual Information Systems

Gerd Stumme[1] and Karl Erich Wolff[2]

[1] Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D–64289 Darmstadt
stumme@mathematik.tu-darmstadt.de
[2] Fachhochschule Darmstadt, Fachbereich Mathematik und Naturwissenschaften
Schöfferstr. 3, D–64295 Darmstadt
wolff@mathematik.tu-darmstadt.de

Abstract. While most data analysis and decision support tools use numerical aspects of the data, Conceptual Information Systems focus on their conceptual structure. This paper discusses how both approaches can be combined.

1 Introduction

The data model of Conceptual Information Systems relies on the insight that concepts are basic units of human thinking, and should hence be activated in data analysis and decision support. The data model is founded on the mathematical theory of *Formal Concept Analysis*. Conceptual Information Systems provide a multi-dimensional conceptually structured view on data stored in relational databases. They are similar to On-Line Analytical Processing (OLAP) tools, but focus on qualitative (i. e. non-numerical) data. The management system TOSCANA visualizes arbitrary combinations of conceptual hierarchies and allows on-line interaction with the database to analyze and explore data conceptually.

Data tables are usually equipped with different types of structures. While most data analysis tools use their numerical structure, Conceptual Information Systems are designed for conceptually structuring data. As concepts are the basic units of human thought, the resulting data model is quite universal — and is also able to cover numerical aspects of the data. However, up to now, the model does not have any features which support techniques specific to numerical data.

Many applications indicate the need for not only using tools which operate only on numerical or only on conceptual aspects, but to provide an integrative approach combining both numerical and conceptual structures for data analysis and decision support in one tool. In this paper we discuss how the data model of Conceptual Information Systems can be extended by numerical aspects. The developments discussed in the sequel arose mostly from scientific and commercial applications, but for sake of simplicity, we start with a small demonstration application: a Conceptual Information System for a private bank account. But first, we provide some basics about Formal Concept Analysis.

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 117–129, 1999.

	greenhouseeffect	ozone depletion potential	acidification	nutrification	human toxicity potential	carcinogenity/toxicity	ecotoxicity
NOx	×		×	×	×		×
N2O	×	×			×		
NH3			×	×		×	×
SO2	×		×		×		×
CO	×				×		×
CO2	×						

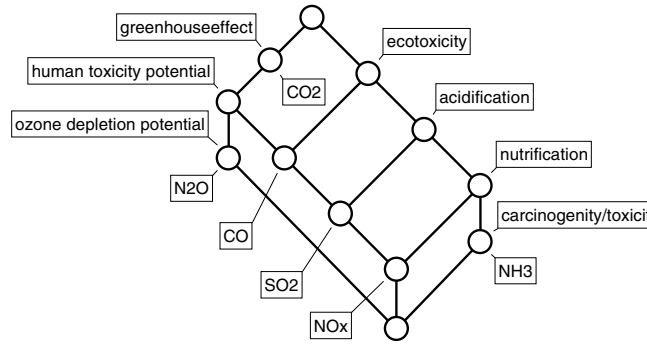


Fig. 1. Formal context and concept lattice of gaseous pollutants

2 The Mathematical Background: Formal Concept Analysis

Concepts are necessary for expressing human knowledge. Therefore, the process of knowledge discovery in databases benefits from a comprehensive formalization of concepts which can be activated to represent knowledge coded in databases. *Formal Concept Analysis* ([10], [1], [13]) offers such a formalization by mathematizing concepts which are understood as units of thought constituted by their extension and intension. For allowing a mathematical description of extensions and intensions, Formal Concept Analysis always starts with a *formal context*.

Definition A *formal context* is a triple (G, M, I) where G is a set whose elements are called (*formal*) *objects*, M is a set whose elements are called (*formal*) *attributes*, and I is a binary relation between G and M (i. e. $I \subseteq G \times M$); in general, $(g, m) \in I$ is read: "the object g *has* the attribute m".

A *formal concept* of a formal context (G, M, I) is defined as a pair (A, B) with $A \subseteq G$ and $B \subseteq M$ such that (A, B) is maximal with the property $A \times B \subseteq I$; the sets A and B are called the *extent* and the *intent* of the formal concept (A, B). The *subconcept-superconcept-relation* is formalized by $(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2 \quad (\Longleftrightarrow B_1 \supseteq B_2)$. The set of all concepts of a context (G, M, I) together with the order relation $\leq$ is always a complete lattice, called the *concept lattice* of (G, M, I) and denoted by $\mathcal{B}(\mathcal{G}, \mathcal{M}, \mathcal{I})$.

Example. Figure 1 shows a formal context about the potential of gaseous pollutants. The six gases NO_x, ..., CO_2 are the objects, and the seven listed perils are the attributes of the formal context. In the line diagram of the concept lattice, we label, for each object $g \in G$, the smallest concept having g in its extent with the name of the object and, for each attribute $m \in M$, the largest concept having m in its intent with the name of the attribute. This labeling

no.	value	paid for	objective	health	date
3	42.00	Konni	ski-club	s	03. 01. 1995
20	641.26	Konni, Florian	office chairs	/	23. 01. 1995
27	68.57	Family	health insurance	hi	02. 02. 1995
34	688.85	Tobias	office table	/	06. 02. 1995
37	25.00	Father	gymn. club	s	08. 02. 1995
52	75.00	Konni	gymn. club	s	24. 02. 1995
73	578.60	Mother	Dr. Schmidt	d	10. 03. 1995
77	45.02	Tobias	Dr. Gram	d	17. 03. 1995
80	77.34	Parents	money due	/	21. 03. 1995

Fig. 2. Withdrawals from a private bank account

allows us to determine for each concept its extent and its intent: The extent [intent] of a concept contains all objects [attributes] whose object concepts [attribute concepts] can be reached from the concept on a descending [ascending] path of straight line segments. For instance, the concept labeled with CO has {CO, SO_2, NO_x} as extent, and {human toxicity potential, greenhouse effect, ecotoxicity} as intent. The concept lattice combines the view of different pollution scenarios with the influence of individual pollutants. Such an integrated view can be of interest for the planning of chimneys for plants generating specific pollutants.

In the following, we distinguish, for each formal concept $\rfloor$, between its *extent* (i. e., the set of all objects belonging to $\rfloor$) and its *contingent* (i. e., the set of all objects belonging to $\rfloor$ but not to any proper subconcept of $\rfloor$). In the standard line diagram, the contingent of a formal concept $\rfloor$ is the set of objects which is represented just below the point representing $\rfloor$. The extent of the largest concept is always the set of all objects. The extent of an arbitrary concept is exactly the union of all contingents of its subconcepts.

In many applications, the data table does not only allow Boolean attributes as in Fig. 1, but also many-valued attributes. In the next section, we show by means of an example how such *many-valued contexts* are handled by formal concept analysis.

3 The Conceptual Aspect of the Bank Account System

The basic example underlying this paper consists of a table of all withdrawals from a private bank account during several months. A small part of this table is shown in Fig. 2. As an example, the row numbered 20 contains information about a withdrawal of 641.26 DM for office chairs for the sons Konni and Florian paid on January 23, 1995. In formal concept analysis, data tables such as the one in Fig. 2 are formalized as *many-valued contexts*.

	sport	doctor	health insurance	health
health="s"	×			×
health="d"		×		×
health="hi"			×	×
health="/"				

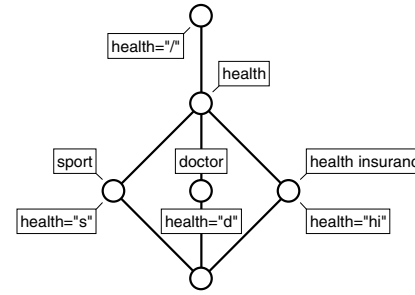


Fig. 3. The scale "health"

Definition A *many-valued context* is a tuple $K := (G, M, (W_m)_{m\in M}, I)$, where G, M, and W_m, $m \in M$, are sets, and $I \subseteq \{(g, m, w) \,|\, g\in G, m\in M, w\in W_m\}$ is a relation where $(g, m, w_1) \in I$ and $(g, m, w_2) \in I$ implies $w_1=w_2$. Thus, each $m \in M$ can be seen as a partial function. For $(g, m, w) \in I$ we say that "object g has value w for attribute m" and write $m(g) = w$.

Clearly each finite many-valued context can be represented as a relational database table where the set G of objects occurs in the first field chosen as primary key.

In the following we construct a conceptual overview with the purpose to answer questions like "How much has been paid for health for each family member?". Therefore we first introduce a conceptual language representing the meaning of the values occuring in the column health of Fig. 2. This language is represented by the formal context in Fig. 3. For example, the withdrawals labeled "s" in column health of Fig. 2 are assigned to "sport" and to "health", while the withdrawals labeled "/" are not assigned to any attribute of this scale. The concept lattice of this formal context is represented by the line diagram in Fig. 3 which demonstrates graphically the intended distinction between the withdrawals not assigned to "health" and those assigned to "health" and the classification of these into three classes. This is an example of a conceptual scale in the sense of the following definition.

Definition A *conceptual scale* for an attribute $m \in M$ of a many-valued context $(G, M, (W_m)_{m\in M}, I)$ is a formal context $S_m := (W_m, M_m, I_m)$.

Conceptual scales serve for "embedding" the values of a many-valued attribute in a conceptual framework describing the aspects the user is interested in. But to embed also the original objects, in our example the withdrawals, into this framework we have to combine the partial mapping of the many-valued attribute m and the embedding of the values. This is done in the following definition of a realized scale.

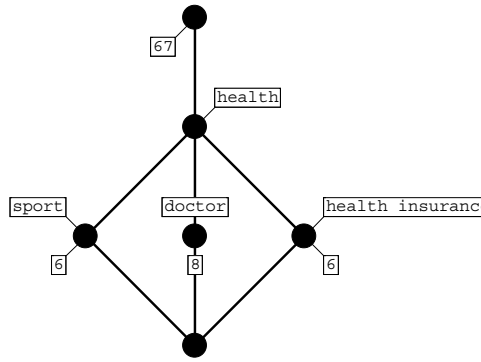


Fig. 4. Frequencies of withdrawals related to "health".

Definition Let $S_m = (W_m, M_m, I_m)$ be a conceptual scale of an attribute m of a many-valued context $(G, M, (W_m)_{m \in M}, I)$. The context (G, M_m, J) with $gJn :\Longleftrightarrow \exists w \in W_m : (g, m, w) \in I \wedge (w, n) \in I_m$ is called the *realized scale* for the attribute m.

To construct the concept lattice of the realized scale we assign to each value w of m, hence to each object of the scale S_m, an SQL-query searching for all objects g in the given many-valued context such that $m(g) = w$. The concept lattice of the realized scale for "health" is shown in Fig. 4 where the contingents are replaced by their cardinalities, called *frequencies*.

Reading example: There are exactly six withdrawals assigned to "sport" and exactly 67 withdrawals not assigned to "health". Finally we remark that there are no withdrawals which are assigned to "health" but neither to "sport", "doctor" or "health insurance".

In TOSCANA, the user can choose conceptual scales from a menu. The database is queried by SQL-statements for determining the contingents of the concepts. Finally the results are displayed in a line diagram representing the embedding of the concept lattice of the realized scale in the concept lattice of the scale.

The line diagram in Fig. 4 is unsatisfactory insofar as we would like to see not only the frequencies of withdrawals but the amount of money paid. In the next section we shall discuss how this can be visualized.

4 The Numerical Aspect of the Bank Account System

For an efficient control of the household budget, the user needs an overview over the distribution of the money, and not of the number of withdrawals. Hence, for each contingent S, we display the sum over the corresponding entries in the column "value" instead of the frequency of S. The result of this computation is

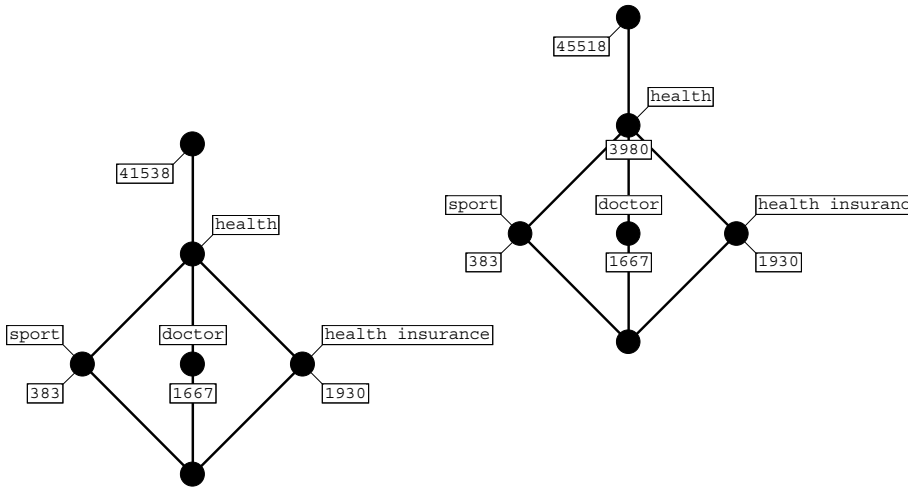


Fig. 5. Summing up book-values over contingents (left) and extents.

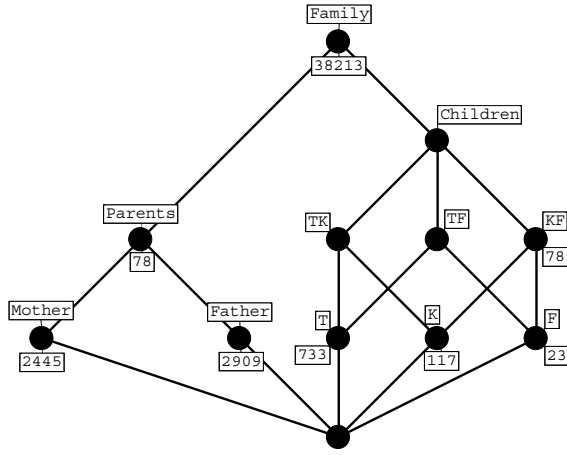


Fig. 6. Summing up book-values over contingents of the scale "family".

the left line diagram in Fig. 5. We can see for example that the withdrawals for "sport" sum up to 383 DM and the withdrawals not concerning "health" sum up to 41538 DM. The right line diagram shows, for each formal concept, the sum over the values of all withdrawals in the extent (instead of the contingent) of this concept, for instance the total amount of 45518 DM for all withdrawals in the given data table and the amount of 3980 DM for "health". To visualize also the amount of money paid for the family members (and for relevant groups of them), we use the scale "family" in Fig. 6. This diagram shows for instance

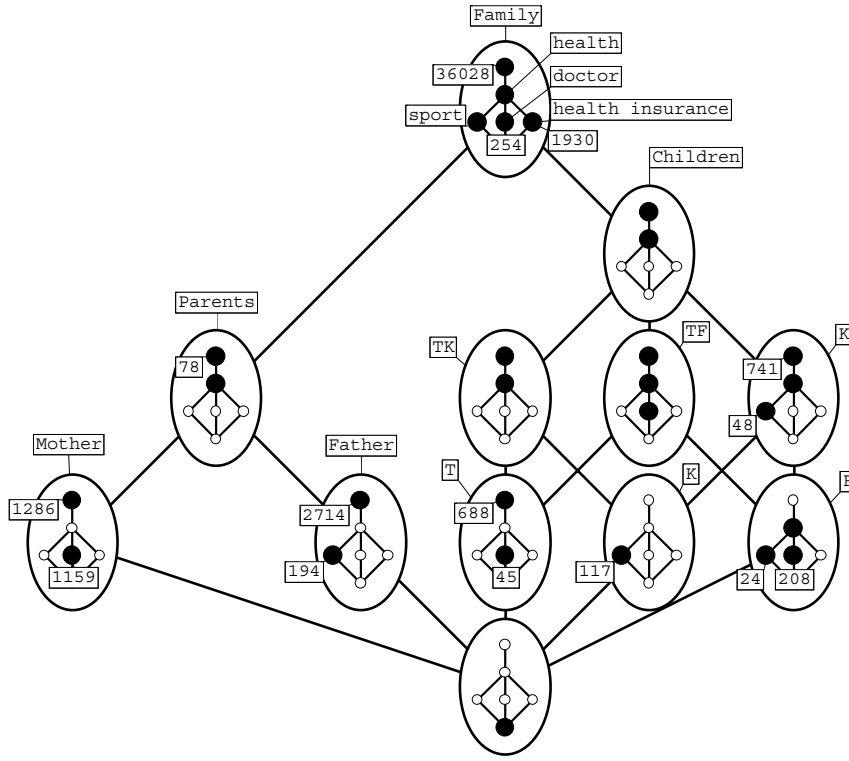


Fig. 7. Summing up over contingents in the nested line diagram of the scales "health" and "family".

that there are withdrawals of 2445 DM for "Mother", that 78 DM are classified under "Parents", but not under "Mother" or "Father" (this is the "money due" in the last row of Fig. 2) and that 38213 DM appear for withdrawals classified under "Family" which are not specified further.

Next we combine the scales "family" and "health". The resulting *nested line diagram* is shown in Fig. 7. Now the withdrawals are classified with respect to the direct product of the scales for family and health. For instance, 733 DM expended for Tobias split into 45 DM for a doctor and 688 DM not concerning health, i. e., the amount spent on his office table (see Fig. 2). This nested line diagram shows also that the withdrawals for health insurance (which amount to 1930 DM) are all summarized under the concept "Family" and are not specified further.

In the next section, we describe the formalization of numerical structures. This is the basis for generalizing the example in Section 6.

5 Relational Structures

In Section 2, we have seen how conceptual structures are formalized. Let us now consider the numerical aspect of the data. In fact, the formalization is a bit more general, such that it covers arbitrary relations and functions on arbitrary sets. It is based on the mathematical notion of *relational structures*.

In the bank account example, the bankbook values are real numbers, for which addition is defined. In general, for each $m \in M$, there may be functions and relations on the set W_m.

Definition A *relational structure* $R := (W, \mathcal{R}, \mathcal{F})$ consists of a set W, a set $\mathcal{R}$ of relations $R \subseteq W^{ar(R)}$ on W, and a set $\mathcal{F}$ of functions $f: W^{ar(f)} \to W$, where ar assigns to each relation and function its arity.

For instance, the data types implemented in the database management system (e. g., `Integer`, `Real`, `Boolean`, `Currency`, or `Datetime`) are relational structures. Hence, for each attribute $m \in M$, we can capture the algebraic structure of its possible attribute values by a relational structure $R_m := (W_m, \mathcal{R}_{\Updownarrow}, \mathcal{F}_{\Updownarrow})$, just as we captured their hierarchical relationships by a conceptual scale S_m.

Definition A *conceptual-relational scheme* of a family $(W_m)_{m \in M}$ of sets is a family $(\mathcal{R}_{\Updownarrow}, \mathcal{S}_{\Updownarrow})_{\Updownarrow \in \mathcal{M}}$ where, for each $m \in M$, $R_m := (W_m, \mathcal{R}_{\Updownarrow}, \mathcal{F}_{\Updownarrow})$ is a relational structure and $S_m = (W_m, M_m, I_m)$ is a conceptual scale.

Here we should mention, that sometimes conceptual and relational aspects overlap. Depending on the purpose, they should be covered by a relational structure or by a conceptual scale, or by both. Time, for instance, can be captured by a linear order in a relational structure or by some scale (e. g., an inter-ordinal scale, if only certain time intervals are of interest).

Relational structures can be used for creating new scales. This *logical scaling* was developed by S. Prediger (cf. [5]). In this paper, however, we discuss only how relational structures may affect the data analysis process once the conceptual scales are created.

6 Conceptual Scaling Supported by Relational Structures

The bank account example and other applications show that it is useful not to analyze numerical and conceptual aspects of the data independently, but to combine them. In this section, we discuss how Conceptual Information Systems can be extended by a numerical component. Since the required functionalities differ from application to application, the idea is to delegate application-specific computations to an external system (e. g., book-keeping system, CAD system, control system, etc.). TOSCANA already provides an SQL-interface to the relational database management system in which the many-valued context is stored,

so that we can use the numerical tools of the relational database system (as, for instance, in the bank account example).

In the process of going from the request of the user to the diagram shown on the screen, we can distinguish two consecutive, intermediary subprocesses. First, the chosen scale is imported from the conceptual scheme, and to each of its concepts, a subset of objects is assigned (by default, its extent or contingent). Second, for each of these sets, some algebraic operations may be performed. Most of the implemented Conceptual Information Systems only activate the first step. Our bank account system is an example where the second step is also activated. In the first step, we also can identify two actions where a numerical component can influence the analysis or retrieval process: the import of scales from the conceptual scheme, where parameters can be assigned to parametrized scales, and the import of objects from the database, which can be sorted out by filters. Finally, we can imagine a further action, following the display of the line diagram, which results in highlighting interesting concepts. These four activities which make an interaction between conceptual and numerical component possible now shall be discussed in detail.

6.1 Adapting Conceptual Scales to the Data

A conceptual scale represents knowledge about the structure of the set W_m of possible values of the attribute m. In general, it is independant from the values $m(G)$ that really appear in the database. In some situations however, it is desirable to construct the scale automatically depending on $m(G)$.

Inter-ordinal scales are typically used when a linear order (e. g., a price scale, a time scale) is divided into intervals with respect to their meaning. The boundaries of the intervals are usually fixed by a knowledge engineer. However, the range of possible attribute values is not always known a priori. Hence, for a first glance at the data, it has proved useful to query the database for the minimal and the maximal value and to split up this interval into intervals of equal length. Depending on the application, it might also be useful to fix the boundaries on certain statistical measures, as for instance average, median, quantiles. These "self-adapting scales" reduce the effort needed to create the conceptual scheme, since they are re-usable. It is planned to implement a user interface by means of which the user can edit parameters at runtime. For instance, he could first invoke an inter-ordinal scale with equidistant boundaries and then fine-tune it according to his needs.

This user interface leads to the second example, an application in control theory: Process data of the incineration plant of Darmstadt were analyzed in order to make the control system more efficient (cf. [2]). Process parameters like `ram velocity` and `steam` are stored in a database. The ram velocity does not influence the steam volume directly, but only with a certain time delay. When the time delay is kept variable, the user can change it via the interface during the runtime of TOSCANA. That can be used, for instance, for determining the time delay of two variables experimentally: The engineer examines the nested line diagram of the corresponding scales for ordinal dependencies. By varying

the shift time, he tries to augment the dependencies, and to determine in this way the time delay.

The possibility of using parameters is also of interest for filters that control the data flow from the database to TOSCANA. They are discussed in the next subsection.

6.2 Filtering the Objects of the Many-Valued Context

In many applications, users are interested in analyzing only a specific subset of objects of the many-valued context; for instance, if one is interested in the withdrawals from the bank account during the past quarter only. If such a subset is determined conceptually, being the extent (more rarely the contingent) of a concept of a suitable combination of conceptual scales, TOSCANA provides for the possibility of "zooming" into that specific concept by mouse click. In the sequel of the analysis only objects belonging to that concept are considered.

But if the interesting subset is not available as extent or contingent of some combinations of earlier constructed scales it is often easier to use a filter. Filters are designed to generate one single interesting subset of the set of objects while conceptual scales generate a whole set of interesting extents and all their intersections and contingents.

For such applications, the conceptual scheme should be extended by *filters*. In addition to conceptual scales, the user can choose filters from a menu. When a filter is activated, then objects are only considered for display if they pass the filter. A filter is realized as an SQL-fragment that is added by AND to the conditions provided by the chosen scales.

The remarks about parameters in the previous subsection apply to filters as well. An example for the use of parameters in filters is again the system of Sects. 2 and 3. As described above, we can construct a filter that only accepts withdrawals effected in a certain period, e. g., the last quarter. The interface for editing parameters introduced in Sect. 5.1 provides the possibility of examining the withdrawals of any period required. When the user activates the filter, he is asked for start and end date.

6.3 Focussing on Specific Aspects of the Objects

The bank account system is an example of focussing on different aspects of the data. There we focus not only on withdrawal numbers, but also on the sum of bankbook values. Now we discuss how this example fits into the formalization described in Sect. 4. Once the user has chosen one or more scales, TOSCANA determines for each concept of the corresponding concept lattice a set S of objects – in most cases its extent or its contingent. In Sect. 1 we mentioned that the user can choose for each concept whether all names of the objects in S shall be displayed or only the cardinality of S. A third standard aspect in TOSCANA is the display of relative frequencies. The last two aspects are examples of algebraic operations.

The focussing in the example of Sects. 2 and 3 can be understood as being composed of two actions: Firstly, instead of working on the set S, the sequence $(m(g) \in W_m)_{g \in S}$ is chosen. In the bank account example, this *projection* assigns to each withdrawal from S the corresponding book-value. Secondly, the sum $\sum(m(S)) := \sum_{g \in S} m(g)$ is computed (and displayed). The latter is done in the relational structure assigned to the corresponding attribute. In TOSCANA, this is realized by a modification of the way the SQL-queries are generated: the standard COUNT-command used for the computation of the frequency of S is replaced by a SUM-command operating on the column "value".

6.4 Highlighting Interesting Concepts

Focussing also can be understood in a different setting. It also means drawing the user's attention to those concepts where the frequency of objects (or the sum of book-values, etc.) is extraordinarily high (or low). The determination of these concepts is based on the frequency distribution of the nested line diagram. This distribution can be represented – without its conceptual order – by a contingency table with entry n_{ij} in cell (i, j) where i (j, resp.) is an object concept of the first (second) scale. As a refinement of Pearson's Chi- Square calculations for contingency tables we recommend calculating for each cell (i, j) the *expected frequency* $e_{ij} := (n_i n_j)/n$ ("expected" means "expected under independence assumption") where n_i (n_j, resp.) is the frequency of object concept i (j, resp.) and n is the total number of all objects. To compare the distribution of the observed frequencies n_{ij} and the expected frequencies e_{ij}, one should study the *dependency double matrix* (n_{ij}, e_{ij}). Pearson's Chi-Square calculations reduce the dependency double matrix to the famous $\chi^2 := \sum_{ij}((n_{ij} - e_{ij})^2/e_{ij})$. But the matrix also can be used as a whole in order to highlight interesting places in a nested line diagram:

- If the user wants to examine the dependency double matrix in detail, then he may choose to display their entries at the corresponding concepts. Additionally, one of the matrices of differences $n_{ij} - e_{ij}$, quotients $(n_{ij} - e_{ij})/e_{ij}$, or quotients n_{ij}/e_{ij} may be displayed in the same way. The conceptual structure represented by the line diagram helps us to understand the dependency double matrix.
- If a less detailed view is required, then the calculation component can generate graphical marks which indicate those concepts where the matrix entries are above or below a given threshold. A typical condition in applications is "$e_{ij} > k$ and $n_{ij}/e_{ij} > p$" where k and p are parameters which can be chosen on a suitable scale.

The Chi-Square formula is a very rough reduction of the information about dependencies, but, clearly, the degree of reduction depends on the purpose of the investigation. If one is interested not only in having an index showing whether there is a dependency, but in understanding the dependencies between two many-valued attributes with respect to chosen scales in detail, then one should carefully

study the distribution of observed and expected frequencies. This can be done with the program DEPEND developed by C. Wehrle in his diploma thesis ([9], supervised by K. E. Wolff).)

7 Outlook

The connections between conceptual scales and relational structures should be studied extensively. Therefore, practical relevant examples containing both parts should be considered.

It is of particular interest to examine the compatibility of various conceptual scales and relational structures on the same set of attribute values. From a formal point of view both structures are of the same generality in the sense that each conceptual scale can be described as a relational structure and vice versa. But they are used differently: conceptual scales generate overviews for knowledge landscapes, while relational structures serve for computations.

This paper discussed how numerical components can support conceptual data processing. One should also investigate how, vice-versa, results of data analysis and retrieval activities in Conceptual Information Systems can be made accessible to other systems. This discussion may lead to hybrid knowledge systems composed of conceptual, numerical and also logical subsystems, each focussing on different aspects of the knowledge landscape inherent in the data.

References

1. B. Ganter, R. Wille: Formale Begriffsanalyse: Mathematische Grundlagen. Springer, Heidelberg 1996 (English translation to appear) 118
2. E. Kalix: Entwicklung von Regelungskonzepten für thermische Abfallbehandlungsanlagen. TH Darmstadt 1997 125
3. W. Kollewe, C. Sander, R. Schmiede, R. Wille: TOSCANA als Instrument der bibliothekarischen Sacherschließung. In: H. Havekost, H.-J. Wätjen (eds.): *Aufbau und Erschließung begrifflicher Datenbanken.* (BIS)-Verlag, Oldenburg 1995, 95–114
4. W. Kollewe, M. Skorsky, F. Vogt, R. Wille: TOSCANA — ein Werkzeug zur begrifflichen Analyse und Erkundung von Daten. In: R. Wille, M. Zickwolff (eds.): *Begriffliche Wissensverarbeitung — Grundfragen und Aufgaben.* B. I.–Wissenschaftsverlag, Mannheim 1994
5. S. Prediger: Logical scaling in formal concept analysis. LNAI **1257**, Springer, Berlin 124
6. P. Scheich, M. Skorsky, F. Vogt, C. Wachter, R. Wille: Conceptual data systems. In: O. Opitz, B. Lausen, R. Klar (eds.): *Information and classification.* Springer, Heidelberg 1993, 72–84
7. F. Vogt, C. Wachter, R. Wille: Data analysis based on a conceptual file. In: H.-H. Bock, P. Ihm (eds.): *Classification, data analysis, and knowledge organization.* Springer, Heidelberg 1991, 131–140
8. F. Vogt, R. Wille: TOSCANA — A graphical tool for analyzing and exploring data. LNCS **894**, Springer, Heidelberg 1995, 226–233

9. C. Wehrle: *Abhängigkeitsuntersuchungen in mehrwertigen Kontexten*. Diplomarbeit, Fachhochschule Darmstadt 1997 128
10. R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets*. Reidel, Dordrecht–Boston 1982, 445–470 118
11. R. Wille: Lattices in data analysis: how to draw them with a computer In: I. Rival (ed.): *Algorithms and order*. Kluwer, Dordrecht–Boston 1989, 33–58
12. R. Wille: Conceptual landscapes of knowledge: A pragmatic paradigm of knowledge processing. In: *Proc. KRUSE '97*, Vancouver, Kanada, 11.–13. 8. 1997, 2–14
13. K. E. Wolff: A first course in formal concept analysis – How to understand line diagrams. In: F. Faulbaum (ed.): *SoftStat '93, Advances in statistical software* **4**, Gustav Fischer Verlag, Stuttgart 1993, 429–438 118

Nested Data Cubes for OLAP
(extended abstract)

Stijn Dekeyser, Bart Kuijpers*, Jan Paredaens, and Jef Wijsen**

University of Antwerp (UIA), Dept. Math. & Computer Sci.
Universiteitsplein 1, B-2610 Antwerp, Belgium
{dekeyser,kuijpers,pareda,jwijsen}@uia.ua.ac.be

Abstract. *Nested data cubes* (NDCs in short) are a generalization of other OLAP models such as f-tables [4] and hypercubes [2], but also of classical structures as sets, bags, and relations. This model adds to the previous models flexibility in viewing the data, in that it allows for the assignment of priorities to the different dimensions of the multidimensional OLAP data.
We also present an algebra in which most typical OLAP analysis and navigation operations can be formulated. We present a number of algebraic operators that work on nested data cubes and that preserve the functional dependency between the dimensional coordinates of the data cube and the factual data in it. We show how these operations can be applied to sub-NDCs at any depth, and also show that the NDC algebra can express the SPJR algebra [1] of the relational model. Importantly, we show that the NDC algebra primitives can be implemented by linear time algorithms.

1 Introduction and Motivation

Since the 1993 seminal paper of Codd et al [6], *on-line analytical processing* (OLAP) is recognized as a promising approach for the analysis and navigation of data warehouses and multidimensional data [5,7,12,13,16,28]. Multidimensional databases typically are large collections of enterprise data which are arranged according to different dimensions (e.g., time, measures, products, geographical regions) to facilitate sophisticated analysis and navigation for decision support in, for instance, marketing. Figure 1 depicts a three-dimensional database, containing sales information of stores. A popular way of representing such information is the "data cube" [9,17,28]. Each dimension is assigned to an axis in n-dimensional space, and the numeric values are placed in the corresponding cells of the 'cube.'

The effectiveness of analysis and the ease of navigation is mainly determined by the flexibility that the system offers to rearrange the perspectives on different dimensions and by its ability to efficiently calculate and store summary information. A sales manager might want to look at the sales per store, or he might

* Post-doctoral research fellow of the Fund for Scientific Research of Flanders (FWO).
** Affiliated to the University of Brussels (VUB) at the time this research was done.

Y. Kambayashi et al. (Eds.): ER'98, LNCS 1552, pp. 129–141, 1999.

Day : day	*Item* : item	*Store* : store	
Jan1	Lego	Navona	→ 32
Jan1	Lego	Colosseum	→ 24
Jan1	Scrabble	Navona	→ 13
Jan1	Scrabble	Colosseum	→ 14
Jan1	Scrabble	Kinderdroom	→ 22
Jan2	Lego	Kinderdroom	→ 2
Jan2	Lego	Colosseum	→ 21

Fig. 1. A multidimensional database.

want to see the total number of items sold in all his stores in a particular month. OLAP systems aim to offer these qualities.

During the past few years, many commercial systems have appeared that offer, through ever more efficient implementations, a wide number of analysis capabilities. A few well-known examples are Arbor Software's *Essbase* [10], IBM's *Intelligent Server* [19], Red Brick's *Red Brick Warehouse* [24], Pilot Software's *Pilot Decision Support Suite* [23], and Oracle's *Sales Analyzer* [25]. Some of these implementations are founded on theoretical results on efficient array manipulation [21,22,26].

In more recent years, however, it has become apparant that there is a need for a formal OLAP model that explicitly incorporates the notion of different and independent views on dimensions and also offers a logical way to compute summary information. Lately, a number of starting points for such models have been proposed (an overview is given in [3]). Gyssens et al. [14] have proposed the first theoretical foundation for OLAP systems: the *tabular database* model. They give a complete algebraic language for querying and restructuring two-dimensional tables. Agrawal et al. [2] have introduced a *hypercube* based data model with a number of operations that can be easily inserted into SQL. Cabibbo and Torlone [4] have recently proposed a data model that forms a logical counterpart of multidimensional arrays. Their model is based on dimensions and *f-tables*. Dimensions are partially ordered categories that correspond to different ways of looking at the multidimensional information (see also [9]). F-tables are structures to store factual data functionally dependent on the dimensions (such as the one depicted in Figure 1). They also propose a calculus to query f-tables that supports multidimensional data analysis and aggregation.

In this paper, we generalize the notions of f-tables and hypercube by introducing the *nested data cube model*. Our data model supports a variety of nested versions of f-tables, each of which corresponds to an assignment of priorities to the different dimensions. The answer to the query *"Give an overview of the stores and their areas together with the global sales of the various items,"* on the data cube of Figure 1, for instance, is depicted in Figure 2 by a nested data cube. We also present an *algebra* to formulate queries, such as the one above, on nested data cubes. This query language supports all the important OLAP analysis and navigational restructuring operations on data cubes. There are a number of operations whose aim lies purely in rearranging the information in the cube in order to create different and independent views on the data. These

include, for instance, nesting and unnesting operations, factual data collection in bags, duplication, extention of the cube with additional dimensions, and renaming. The algebra also contains selection, aggregation, and roll-up operations, which support data analysis.

Motivation. The model we propose offers a natural paradigm for perceiving large data cubes, as it adds to previously mentioned models mainly flexibility in viewing the data by assigning priorities to different dimensions. Put another way, grouping on values of an attribute is made explicit. As is shown in the next section, the NDC model can also be used to represent common data structures such as sets, bags, and relations. Furthermore, our model generalizes and improves upon a number of other OLAP models, as discussed later in this paragraph.

Let us first come back to the problem of perceiving large data cubes. Traditional data cubes are "flat" in the sense that they treat each dimension in the same way. This causes two kinds of perceptional difficulties. Firstly, the *dimensionality* can be too high to be practically visualizable in a 'cubic' format. Secondly, the *cardinality* (the number of tuples in the database) is typically very high. Our approach can be used to decrease both measures.

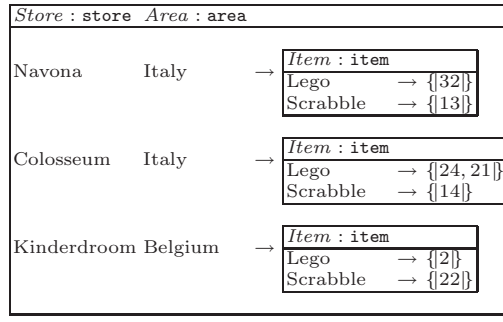


Fig. 2. A three-dimensional data cube containing sales information.

We now turn to the comparison of our approach to other OLAP models. While the query language for the tabular data model proposed by Gyssens et al. [14] only covers restructuring of tables, ours supports both restructuring and complex analysis queries. Their model, although generalized for an arbitrary number of dimensions [15], is mainly suited for two-dimensional spreadsheet applications. The NDC model, however, offers a theoretical framework for "cube viewers" where users navigate through a space of linearly nested n-dimensional cubes.

Our model is based on a simple hypercube model such as the one proposed by Agrawal et al. [2]. Their approach primarily deals with the insertion of their alebra into SQL, while this paper proposes an independent implementation of nested data cubes.

The NDC model most closely resembles the f-tables proposed by Cabibbo et al. [4]. Like their model, ours contains explicit notions of dimensions and describes hierarchies of levels within dimensions in a clean way. However, the model we propose also allows the construction of a hierarchy of the dimensions in an NDC. This does not only make viewing very large cubes easier, it also gives semantics to the scheme of a data cube. Different end-users will typically prefer different schemes for the same underlying data.

An important problem with the calculus for f-tables as proposed by Cabibbo et al. is that in at least two cases it is necessary to leave the model temporarily. Firstly, when aggregate functions are used, the result of a query may no longer be functional. In contrast, the NDC model has the ability to collect factual data in bags which allows us to stay within our model after grouping and before aggregation. Once data is collected in bags, a wide variety of aggregate functions can be performed on them by reusing the constructed bags.

Secondly, in the f-table model, it is not clear where in the system the information for the roll-up function is to be found. It is assumed that it is known for every value in any level how to roll-up to a value in a higher level from the hierarchy. Conversely, our operator that is used for roll-up takes any relation as input, meaning that roll-up information can be stored in another NDC.

While our model clearly shares similarities with the nested relational model [11,18,27], it is not a generalization of it. Importantly, our model imposes linear nesting which only allows for the construction of a linear hierarchy of dimensions. Finally, Lehner [20] considered a Nested Multidimensional Data Model where nesting, however, mainly applies to the modeling of dimensional levels. Furthermore, their approach does not formalize the content of the cube and permits only single numerical values as cube entries. Also, it does not allow for user defined aggregation, nor does it treat dimensions and measures symmetrically.

Organization. Section 2 introduces *nested data cubes* (NDCs). Section 3 introduces the operators of the NDC algebra and illustrates them by presenting an extensive example. Section 4 contains two results concerning the expressive power of the NDC algebra. Section 5 shows how our algebra can be implemented efficiently. For formal definitions and the proofs of the theorems, we refer to [8].

2 Nested Data Cubes

In this section, we formally define the nested data cube model and illustrate the definitions using the data cube of Figure 2. After giving additional examples, we show that the set of NDCs over a given scheme is recursively enumerable. Algebraic operators that work on NDCs are presented in the next section.

In what follows, we use the delimiters $\{|\cdot|\}$ to denote a bag. We assume the existence of a set $\mathcal{A}$ of *attributes* and a set $\mathcal{L}$ of *levels*. In Figure 2, the attributes are *Store*, *Area*, and *Item*, and the levels are `store`, `area`, `item`, and `num` (the set of natural numbers). Every level l has a recursively enumerable set $dom(l)$

of atomic values associated to it. For technical reasons, the set $\mathcal{L}$ contains a reserved level, λ, which has a singleton domain: $dom(\lambda) = \{\top\}$, with $\top$ the Boolean true value. We define $\mathbf{dom} = \bigcup\{dom(l) \mid l \in \mathcal{L}\}$.

For certain pairs (l_1, l_2) of $\mathcal{L} \times \mathcal{L}$, there exist a *roll-up function*, denoted $\texttt{R-UP}_{l_1}^{l_2}$, that maps every element of l_1 to an element of l_2. Further requirements may be imposed on the nature of roll-up functions, as is done in [4].

Definition 1. (Coordinate).

- A *coordinate type* is a set $\{A_1 : l_1, \ldots, A_n : l_n\}$ where $A_1, \ldots, A_n$ are distinct attributes, $l_1, \ldots, l_n$ are levels, and $n \geq 0$.
- A *coordinate* over the coordinate type $\{A_1 : l_1, \ldots, A_n : l_n\}$ is a set $\{A_1 : v_1, \ldots, A_n : v_n\}$ where $v_i \in dom(l_i)$, for $1 \leq i \leq n$.

The set of attributes appearing in a coordinate (type) γ, is denoted $att(\gamma)$. □

The NDC of Figure 2 has two coordinate types; i.e., $\{Store : \texttt{store}, Area : \texttt{area}\}$ and $\{Item : \texttt{item}\}$. An example of a coordinate over the former coordinate type is $\{Store :$ Navona, $Area :$ Italy$\}$.

Definition 2. (Scheme). The abstract syntax of a *nested data cube scheme* (NDC *scheme*, or simply *scheme*) is given by:

$$\tau = [\delta \to \tau] \quad | \quad \beta \tag{1}$$

$$\beta = l \quad | \quad \{\!|\beta|\!\} \tag{2}$$

where δ is a coordinate type, and l is a level. Throughout this paper, the Greek characters δ, τ, and β consistently refer to the above syntax. □

The nested data cube of Figure 2, for example, is

$$\tau_0 = [\{Store : \texttt{store}, Area : \texttt{area}\} \to [\{Item : \texttt{item}\} \to \{\!|\texttt{num}|\!\}]].$$

As another example, the scheme of Figure 1 is $[\{Day : \texttt{day}, Item : \texttt{item}, Store : \texttt{store}\} \to \texttt{num}]$.

Definition 3. (Instance). To define an instance (*nested data cube*) over a scheme τ, we first define the function $dom(\cdot)$ as follows:

$$\begin{aligned}
dom(\delta) &= \text{the set of all coordinates over coordinate type } \delta \\
dom([\delta \to \tau]) &= \{\{v_1 \to w_1, \ldots, v_m \to w_m\} \mid m \geq 0, \text{ and} \\
&\qquad v_1, \ldots, v_m \text{ are pairwise distinct coordinates of } dom(\delta), \text{ and} \\
&\qquad w_i \in dom(\tau) \text{ for } 1 \leq i \leq m\} \\
dom(\{\!|\beta|\!\}) &= \{\{\!|v_1, \ldots, v_m|\!\} \mid v_i \in dom(\beta) \text{ for } 1 \leq i \leq m\}
\end{aligned}$$

An NDC over the scheme τ is an element of $dom(\tau)$. □

Figure 2 is a representation of an instance of the NDC with scheme τ_0.

Definition 4. (Depth). The *depth* of a scheme τ, denoted $depth(\tau)$, is defined as the number of occurences of δ in the construction of τ by applying rule (1) of Definition 2.

The notion of depth is extended to NDCs in an obvious way: if C is an NDC over the scheme τ, then we say that the depth of C is $depth(\tau)$. The notion of *subscheme* of a scheme τ at depth n is assumed to be intuitively clear. □

The NDC with scheme τ_0 is of depth 2. The subscheme at depth 2 is $[\{Item : \texttt{item}\} \rightarrow \{\!|\texttt{num}|\!\}]$.

We now give some additional examples.

Example 1. Note that $[\{\} \rightarrow \texttt{num}]$ is a legal scheme. Its depth is equal to 1. All NDCs over this scheme can be listed as follows (assume $dom(\texttt{num}) = \{1, 2, \ldots\}$): $\{\}$ (the empty NDC), $\{\{\} \rightarrow 1\}$, $\{\{\} \rightarrow 2\}$, and so on.

Importantly, `num` itself is also a legal scheme. Its depth is equal to 0. The NDCs over `num` (as a scheme) are 1,2, ... □

Example 2. NDCs can represent several common data structures, as follows.

Bag: An NDC over a scheme of the form $\{\!|\beta|\!\}$.
Set: An NDC over a scheme of the form $[\{A : l\} \rightarrow \lambda]$.
Relation: An NDC over a scheme of the form $[\delta \rightarrow \lambda]$. The NDC called C_0 in Section 3 represents a conventional relation.
F-tables [4]: An NDC over a scheme of the form $[\delta \rightarrow l]$. □

Example 1 shows that the NDCs over the scheme $[\{\} \rightarrow \texttt{num}]$ are recursively enumerable. The following theorem generalizes this result for arbitrary schemes.

Theorem 1. *The set of all* NDC*s over a given scheme τ is recursively enumerable.*

3 The NDC Algebra

The NDC *algebra* consists of the following eight operators:

`bagify` This operator decreases the depth of an NDC by replacing each innermost sub-NDC by a bag containing the right-hand values appearing in the sub-NDC;

`extend` This operator adds an attribute to an NDC. The attribute values of the newly added attribute are computed from the coordinates in the original NDC;

`nest` *and* `unnest` These operators capture the classical meaning of nesting and unnesting;

`duplicate` This operator takes an NDC and replaces the right-hand values by the attribute values of some specified attribute;

`select` *and* `rename` These operators correspond to operators with the same name in the conventional relational algebra;

`aggregate` This operator replaces each right-hand value w in an NDC by a new value obtained by applying a specified function to w.

Furthermore, the NDC algebra also allows for the use of these operators at arbitrary depths. For instance, the use of `nest` at depth 2, will be denoted by $\texttt{nest}^2$. For more details, we refer to Section 4.1.

These operators are illustrated in the following extensive example, which is purely designed to contain all operators (and thus contains redundant steps). Readers interested in the formal definitions of the operators and the details of their semantics, are refered to [8].

Before turning to the example, we make the following remarks. Due to space restrictions, we omit several intermediate NDCs. Also, to make the tables smaller in size, we have used abbreviations for the attributes. It should be noted that the size of the tables printed below is big because we chose to show all sub-cubes at once. However, in interactive cube-viewers, sub-cubes will only "open" when clicked upon, thus reducing the size profoundly. Finally, the reader should understand that the definitions of the operators are formed such that the result of an operation always retains functionality.

The query used in the example is

> *Give an overview per (area, country) pair and per item of the amounts of toys sold over all shops and all time, in the area of Europe.*

We start from raw data in a relation C_0 containing information about toy shops. Typically, such tables contain more than one attribute that can be seen as a measure. In C_0, for instance, both the number of items sold (*So*) as well as the number of damaged or lost items (*Lo*) can serve as a measure.

Da : `day`	*It* : `item`	*St* : `store`	*So* : `num`	*Lo* : `num`	*Co* : `country`	
Jan1	Lego	Colosseum	35	4	Italy	$\rightarrow \top$
Jan1	Lego	Navona	12	1	Italy	$\rightarrow \top$
Jan1	Lego	Kindertuin	31	6	Belgium	$\rightarrow \top$
Jan1	Lego	Toygarden	31	1	USA	$\rightarrow \top$
Jan1	Scrabble	Atomium	11	2	Belgium	$\rightarrow \top$
Jan1	Scrabble	Colosseum	15	2	Italy	$\rightarrow \top$
Jan1	Scrabble	Funtastic	22	0	Canada	$\rightarrow \top$
Jan1	Scrabble	Kindertuin	19	5	Belgium	$\rightarrow \top$
Jan1	Scrabble	Navona	17	3	Italy	$\rightarrow \top$
Jan2	Lego	Kindertuin	42	5	Belgium	$\rightarrow \top$
Jan2	Lego	Navona	28	7	Italy	$\rightarrow \top$

C_0

Cube C_1 is obtained after selecting one attribute (*So*) from C_0 to be used as a measure, i.e., $C_1 = \texttt{duplicate}(C_0, So)$. After chosing this measure for the OLAP analysis, a logical next step would be to remove this and all other measures from the coordinate type of the NDC. This projection can be simulated

in our NDC algebra by the following three steps; first the measures are put in a seperate sub-cube by using the `nest` operator, then the information is collapsed in bags, and finally the aggregate `sum` is computed.

However, to save space, we temporarily leave the measures in the coordinate type. In later steps, they will disappear.

$C_1 = [\{Da : \texttt{day}, It : \texttt{item}, St : \texttt{store}, So : \texttt{num}, Lo : \texttt{num}, Co : \texttt{country}\} \rightarrow \texttt{num}]$.

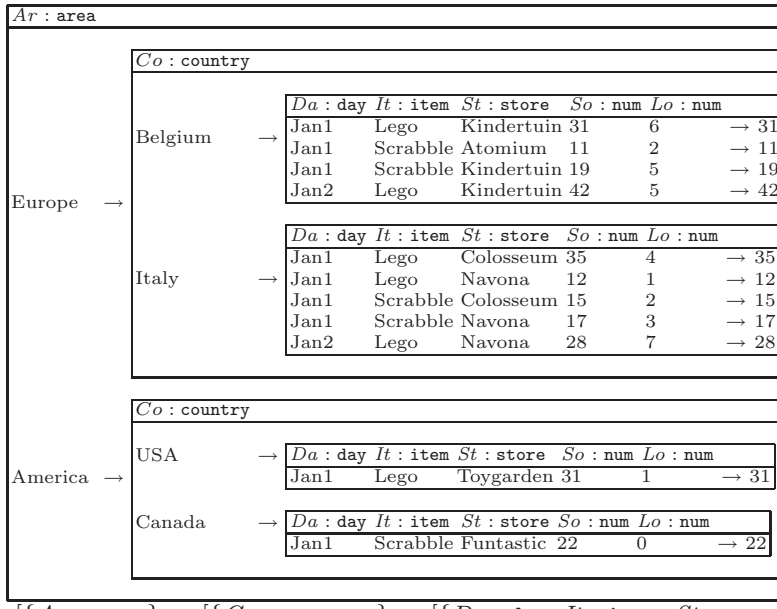
In order to satisfy the query, we have to add the `area` attribute to the coordinate type. This is done by applying `extend`, i.e.,

$C_2 = \texttt{extend}(C_1, Ar, \texttt{area}, \texttt{R-UP}^{\texttt{area}}_{\texttt{country}})$.

Thus, the scheme of C_2 is $[\{Da : \texttt{day}, It : \texttt{item}, St : \texttt{store}, So : \texttt{num}, Lo : \texttt{num}, Co : \texttt{country}, Ar : \texttt{area}\} \rightarrow \texttt{num}]$.

We now nest in two steps. Thus, $C_3 = \texttt{nest}(C_2, Ar)$ and $C_4 = \texttt{nest}^2(C_3, Co)$. The scheme of C_3 then is

$[\{Ar : \texttt{area}\} \rightarrow [\{Da : \texttt{day}, It : \texttt{item}, St : \texttt{store}, So : \texttt{num}, Lo : \texttt{num}, Co : \texttt{country}\} \rightarrow \texttt{num}]]$.



$C_4 = [\{Ar : \texttt{area}\} \rightarrow [\{Co : \texttt{country}\} \rightarrow [\{Da : \texttt{day}, It : \texttt{item}, St : \texttt{store}, So : \texttt{num}, Lo : \texttt{num}\} \rightarrow \texttt{num}]]]$

We now decrease the number of tuples in the cube by performing a selection. Cube $C_5 = \texttt{select}(C_4, Ar, \text{ Europe})$. The scheme of C_5 is the same as that of C_4. To obtain the pairs of (area, country) requested in the query, unnesting is applied to the cube. Cube $C_6 = \texttt{unnest}(C_5)$, and its scheme is

$[\{Ar : \texttt{area}, Co : \texttt{country}\} \rightarrow [\{Da : \texttt{day}, It : \texttt{item}, St : \texttt{store}, So : \texttt{num}, Lo : \texttt{num}\} \rightarrow \texttt{num}]]$.

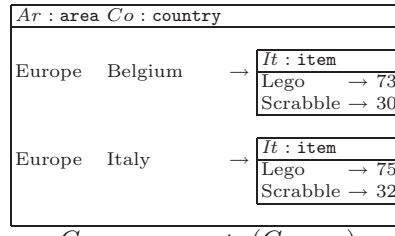
We now need to nest on the `item` attribute (as mentioned in the statement of our example query), appearing in the second level of grouping. Cube $C_7 = \texttt{nest}^2(C_6, It)$, with scheme

$$[\{Ar : \texttt{area}, Co : \texttt{country}\} \rightarrow [\{It : \texttt{item}\} \rightarrow [\{Da : \texttt{day}, St : \texttt{store}, So : \texttt{num}, Lo : \texttt{num}\} \rightarrow \texttt{num}]]].$$

Since we have now obtained the necessary grouping, all attributes remaining at the deepest level of nesting are not needed anymore. Their data is collapsed into bags, i.e., $C_8 = \texttt{bagify}(C_7)$. Note that we are now removing the measures from the coordinate type, and are also putting all information over all dates together.

$$C_8 = [\{Ar : \texttt{area}, Co : \texttt{country}\} \rightarrow [\{It : \texttt{item}\} \rightarrow \{\!|\texttt{num}|\!\}]].$$

As a last step in the process of realizing our example query by using the NDC algebra, we perform the `sum` aggregate on the bags of cube C_8 to obtain the totals of sold items. This yields the desired result.



$$C_9 = \texttt{aggregate}(C_8, \texttt{sum})$$

4 The Expressive Power of the NDC Algebra

In this section, we show some properties concerning the expressiveness of the NDC algebra. We first show that the NDC algebra is sufficiently powerful to capture algebraic operations working directly on sub-NDCs over a subscheme of a scheme. Next, we show that the NDC algebra can express the SPJR algebra [1]. We refer to [8] for the proofs of the theorems in this section.

4.1 Applying Operators at a Certain Depth

The recursion in the definition of NDC is a "tail recursion." Consequently, the "recursion depth" can be used to unequivocally address a sub-NDC within an NDC. This is an interesting and important property of NDCs. It is exploited by defining operators that directly work on sub-NDCs at a certain depth. Such operators reduce the need for frequent nesting and unnesting of NDCs.

Definition 5. Let C be an NDC over the scheme τ. Let $1 \leq d \leq \mathit{depth}(\tau)$. Let $\texttt{op}(C, a_1, \ldots, a_n)$ be any operation of the NDC algebra.

Let $\tau' = \mathit{subscheme}(\tau, d)$. Then $\texttt{op}^d(C, a_1, \ldots, a_n)$ is defined iff $\texttt{op}(\tau', a_1, \ldots, a_n)$ is defined. □

The details of the impact on schemes and NDCs are omitted. They can be found in [8].

In the example of Section 3, cube C_4 was obtained from C_3 by using the `nest` operator at depth 2.

The following theorem states that $\mathtt{op}^d(C, a_1, \ldots, a_n)$ is not a primitive operator; i.e., it can be expressed in terms of the operators of the NDC algebra.

Theorem 2. *Let C be an* NDC *over the scheme τ.*
The operator $\mathtt{op}^d(C, a_1, \ldots, a_n)$ with $d \geq 2$ is redundant.

As an example of this theorem, cube C_4 of the previous section can be obtained from cube C_3 by applying the following expressions at depth 1:

$$C_4 = \mathtt{nest}(\mathtt{nest}(\mathtt{unnest}(C_3), It), Ar).$$

4.2 The SPJR Algebra

Theorem 3. *The* NDC *algebra expresses the SPJR [1] algebra.*

The proof for Theorem 3 (see [8]) shows how the `extend` operator can be used to simulate the relational join.

5 Implementing the NDC Algebra

The operations of Section 3 can be implemented by algorithms that run in linear time with respect to the number of atomic values that appear in the data cube. We assume that each aggregate function is computable in polynomial time.

We introduce two new constructs: the *i*Cube which holds the actual data in an n-dimensional array, and the *i*Struct, essentially a string representing the structure behind the data.

For example, consider the scheme $[\{A_1 : \mathtt{a}_1, A_2 : \mathtt{a}_2\} \rightarrow [\{B_1 : \mathtt{b}_1, B_2 : \mathtt{b}_2\} \rightarrow \mathtt{c}]]$. It can be implemented by the *i*Cube *cube* of type $\mathtt{c}[\#\mathtt{b}_1][\#\mathtt{a}_2][\#\mathtt{d}_1][\#\mathtt{b}_2][\#\mathtt{a}_1]$ (the array type of the Java language is used for simplicity) together with the *i*Struct $[5, 2 \rightarrow [1, 4 \rightarrow \cdot]]$. In the *i*Cube's type, $\#\mathtt{a}_1$ denotes the cardinality of $dom(\mathtt{a}_1)$ plus one. That is, there is one entry for each element of $dom(\mathtt{a}_1)$ (indexes $1, 2, \ldots, \#\mathtt{a}_1 - 1$) on top of the entry with index 0. The numbers in the *i*Struct denote positions in the array type. For example, "5" refers to the fifth dimension, which ranges to $\#\mathtt{a}_1$. A possible member of an NDC over the given scheme is $[\{A_1 : u_1, A_2 : u_2\} \rightarrow [\{B_1 : v_1, B_2 : v_2\} \rightarrow w]]$ which will be represented in the *i*Cube as $cube[v_1][u_2][0][v_2][u_1] = w$.

Note that an extra, unused dimension is present in the *i*Cube (namely, the third dimension ranging to $\#\mathtt{d}_1$). This is necessary in case the `extend` operation is used, as we require the types of *i*Cubes to remain the same throughout the computation of the query. This dimension will then be used to store the attribute created by the `extend` operator.

A final remark relates to the use of bags in our model. The implementation should support fast access to the elements in a bag of an NDC, to facilitate the computation of aggregate functions. Consider, for example, the scheme $[\{A:\mathtt{a}\} \to \{\!|\mathtt{c}|\!\}]$. NDCs over this scheme contain bags. One possible implementation uses the *i*Cube *cube* of type $\mathtt{c}[\#\mathtt{a}][\#\mathtt{d}]$ and the *i*Struct $[1 \to \{\!|2|\!\}]$. A member $[\{A : v\} \to \{\!|w_1, w_2|\!\}]$ of an NDC over this scheme, for example, is represented by $\{\!|cube[v][i] \mid 1 \leq i < \#\mathtt{d}|\!\} = \{\!|w_1, w_2|\!\}$.

The operations of the NDC algebra are implemented in such a way that the type of the *i*Cube never changes. Importantly, the `nest`, `unnest`, and `bagify` operations only change the *i*Struct, leaving the *i*Cube unaffected. The other operations also change the content of the *i*Cube. Based on this, we can implement an expression in the NDC algebra in linear time. We now give a concrete example.

Let *revenue* be an NDC over the scheme

$$[\{Store : \mathtt{store} \to [\{City : \mathtt{city}\} \to [\{Product : \mathtt{product}\} \to \mathtt{num}]]]$$

We want to answer the query *"For each city, give the maximal total revenue realized by any store in that city."* The *i*Cube and initial *i*Struct implementing the NDC are $revenue = \mathtt{num}[\#\mathtt{store}][\#\mathtt{city}][\#\mathtt{product}]$ and $[1 \to [2 \to [3 \to \cdot]]]$, respectively. The algebraic expression for the query is

$$\mathtt{aggregate}(\mathtt{bagify}(\mathtt{aggregate}(\mathtt{bagify}(\mathtt{nest}(\mathtt{unnest}(revenue),)City),)\mathtt{sum}),)\mathtt{max}$$

A Java-like linear program implementing the expression is

```
for (int c = 1; c ≤ #city; c++) {
   revenue[0][c][0] = 0;
   for (int s = 1; s ≤ #store; s++) {
      revenue[s][c][0] = 0;
      for (int p = 1; p ≤ #product; p++) {
         revenue[s][c][0] += revenue[s][c][p];
      }
      revenue[0][c][0] = max(revenue[0][c][0], revenue[s][c][0]);
   }
}
```

References

1. S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995. 129, 137, 138
2. R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proc. IEEE Int. Conf. Data Engineering (ICDE '97)*, pages 232–243, 1997. 129, 130, 131
3. M. Blaschka, C. Sapia, G. Höfling, and B. Dinter. Finding Your Way through Multidimensional Data Models. In *DWDOT Workshop (Dexa 98)*, Vienna, 1998. 130

4. L. Cabibbo and R. Torlone. Querying multidimensional databases. In *Sixth Int. Workshop on Database Programming Languages (DBPL '97)*, pages 253–269, 1997. 129, 130, 132, 133, 134
5. S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1):65–74, 1997. 129
6. E. Codd, S. Codd, and C. Salley. Providing OLAP (On-Line Analytical Processing) to user-analysts: An IT mandate. *Arbor Software White Paper*. 129
7. G. Colliat. Olap, relational, and multidimensional database systems. *SIGMOD Record*, 25(3):64–69, 1996. 129
8. S. Dekeyser, B. Kuijpers, J. Paredaens, and J. Wijsen. Nested Data Cubes. *Technical Report 9804*, University of Antwerp, 1998. `ftp://wins.uia.ac.be/pub/olap/ndc.ps` 132, 135, 137, 138
9. C. Dyreson. Information retrieval from an incomplete data cube. In *Proc. Int. Conf. Very Large Data Bases (VLDB '96)*, pages 532–543, Bombai, India, 1996. 129, 130
10. Essbase. *Arbor Software,* `http://www.arborsoft.com/OLAP.html`. 130
11. P.C. Fischer, and S.J. Thomas. Nested Relational Structures. In *The Theory of Databases, Advances in Computing Research III*, PC. Kanellakis, ed., pages 269–307, JAI Press, Greenwich, CT, 1986. 132
12. J. Gray, A. Boswirth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by. In *Proc. IEEE Int. Conf. Data Engineering (ICDE '97)*, pages 152–159, 1997. 129
13. J. Gray, S. Chaudhuri, A. Boswirth, A. Layman, D. Reichart, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1:29–53, 1997. 129
14. M. Gyssens, L. Lakshmanan, and I. Subramanian. Tables as a paradigm for querying and restructuring. In *Proc. Symposium on Principles of Database Systems (PODS '96)*, pages 93–103, Montreal, Canada, 1996. 130, 131
15. M. Gyssens and L. Lakshmanan. A Foundation for Multi-Dimensional Databases. In *Proc. VLDB '97*, pages 106–115, Athens, Greece, 1997. 131
16. J. Han. OLAP mining: An integration of OLAP with data mining. In *Proceedings of the 7th IFIP 2.6 Working Conf. on Database Semantics (DS-7)*, pages 1–9, 1997. 129
17. V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD '96)*, pages 205–216, Montreal, Canada, 1996. 129
18. G. Jaeschke, and H.-J. Schek. Remarks on the Algebra on Non First Normal Form Relations. In *Proceedings first Symposium on Principles of Database Systems (PODS '82)*, pages 124–138, Los Angeles, CA, 1982. 132
19. Intelligent server. *IBM,* `http://www.software.ibm.com/data/pubs/papers`. 130
20. W. Lehner. Modeling Large Scale OLAP Scenarios. In *Proceedings of EDBT '98*, pages 153–167, Valencia, Spain, 1998. 132
21. L. Libkin, R. Machlin, and L. Wong. A query language for multidimensional arrays: Design implementation, and optimization techniques. In *Proc. Int. Conf. Management of Data (SIGMOD '96)*, pages 228–239, Montreal, Canada, 1996. 130
22. A. Marathe and K. Salem. A language for manipulating arrays. In *Proc. Int. Conf. Very Large Data Bases (VLDB '97)*, pages 46–55, Athens, Greece, 1997. 130
23. Pilot decision support suite. *Pilot Software*, `rickover.pilotsw.com/products/`. 130
24. Red brick warehouse. *Red Brick*, `www.redbrick.com/rbs-g/html/plo.html`. 130

25. Sales analyzer. *Oracle*, `http://www.oracle.com/products/olap/html/`. 130
26. S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. In *Proc. Int. Conf. Data Engineering*, pages 328–336, Houston, TX, 1994. 130
27. H. J. Schek, and M. H. Scholl. The Relational Model with Relation-Valued Attributes. In *Information Systems* **11:2**, pages 137–147, 1986. 132
28. A. Shoshani. OLAP and statistical databases: Similarities and differences. In *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '97)*, pages 185–196, Tucson, AZ, 1997. 129

Author Index