

A Min Tjoa  
Juan Trujillo (Eds.)

LNCS 3589

# Data Warehousing and Knowledge Discovery

7th International Conference, DaWaK 2005  
Copenhagen, Denmark, August 2005  
Proceedings



 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*New York University, NY, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

A Min Tjoa Juan Trujillo (Eds.)

# Data Warehousing and Knowledge Discovery

7th International Conference, DaWaK 2005  
Copenhagen, Denmark, August 22-26, 2005  
Proceedings



Springer

## Volume Editors

A Min Tjoa  
Vienna University of Technology  
Institute of Software Technology and Interactive Systems  
Favoriten Str. 9-11/188, 1040 Vienna, Austria  
E-mail: amin@ifs.tuwien.ac.at

Juan Trujillo  
University of Alicante  
Dept. of Language and Information Systems  
Apto. Correos 99 E-03080, C.P. 03690 Alicante, Spain  
E-mail: jtrujillo@dlsi.ua.es

Library of Congress Control Number: Applied for

CR Subject Classification (1998): H.2, H.3, H.4, C.2, H.5, I.2, J.1

ISSN 0302-9743  
ISBN-10 3-540-28558-X Springer Berlin Heidelberg New York  
ISBN-13 978-3-540-28558-8 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

[springeronline.com](http://springeronline.com)

© Springer-Verlag Berlin Heidelberg 2005  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 11546849 06/3142 5 4 3 2 1 0

## Preface

For more than a decade, data warehousing and knowledge discovery technologies have been developing into key technologies for decision-making processes in companies. Since 1999, due to the relevant role of these technologies in academia and industry, the Data Warehousing and Knowledge Discovery (DaWaK) conference series have become an international forum where both practitioners and researchers share their findings, publish their relevant results and dispute in depth research issues and experiences on data warehousing and knowledge discovery systems and applications.

The 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2005) continued series of successful conferences dedicated to these topics. In this edition, the conference tried to provide the right, logical balance between data warehousing and knowledge discovery. Regarding data warehousing, papers cover different relevant and still unsolved research problems, such as the modelling of ETL processes and integration problems, designing OLAP technologies from XML documents, modelling data warehouses and data mining applications together, improvements in query processing, partitioning and implementations. With regard to data mining, a variety of papers were presented on subjects including data mining techniques, clustering, classification, text documents and classification, and patterns. These proceedings contain the technical papers that were selected for presentation at the conference.

We received 196 abstracts, and finally received 162 papers from 38 countries, and the Program Committee eventually selected 51 papers, making an acceptance rate of 31.4 % of submitted papers.

We would like to express our gratitude to all Program Committee members and external reviewers who reviewed the papers very profoundly and in a timely manner. Due to the high number of submissions and the high quality of the submitted papers, the reviewing, voting and discussion process was an extraordinarily challenging task. Special thanks must be given to Mr. Tho Manh Nguyen for all his support in the organization of the PC-tasks of DaWaK 2005. We would also like to thank all the authors who submitted their papers to DaWaK 2005 as their contributions built the basis of this year's excellent technical program.

Many thanks go to Ms. Gabriela Wagner for providing a great deal of assistance in the administering of the DaWaK management issues as well as to Mr. Raimund Angleitner-Flotzinger and Mr. Andreas Dreiling for their supervision of the conference management software.

August, 2005

A. Min Tjoa  
Juan Trujillo

# Program Committee

## Conference Program Chairpersons

A Min Tjoa, Vienna University of Technology, Austria

Juan C. Trujillo, University of Alicante, Spain

## Program Committee Members

Alberto Abello, Polytechnic University of Catalunya, Spain

Elena Baralis, Polytechnic University of Turin, Italy

Jorge Bernardino, Polytechnic of Coimbra, Portugal

Claudio Bettini, University of Milan, Italy

Sourav S. Bhowmick, Nanyang Technological University, Singapore

Ella Bingham, Helsinki University of Technology, Finland

Mokrane Bouzeghoub, University of Versailles, France

Stephane Bressan, National University of Singapore, Singapore

Peter Brezany, University of Vienna, Austria

Robert M. Bruckner, Microsoft, USA

Luca Cabibbo, Università di Roma Tre, Italy

Chee-Yong Chan, National University of Singapore, Singapore

Arbee L.P. Chen, National Chengchi University, Taipei, Taiwan

Karen Davis, University of Cincinnati, Cincinnati, USA

Vladimir Estivill-Castro, Griffith University, USA

Christie Ezeife, University of Windsor, Canada

Ling Feng, University of Twente, Netherlands

Eduardo Ferrandez-Medina, University of Castilla La Mancha, Spain

Jean-Gabriel Ganascia, Université Pierre et Marie Curie, France

Matteo Golfarelli, University of Bologna, Italy

Jerzy Grzymala-Busse, University of Kansas, USA

S.K. Gupta, Indian Institute of Technology, India

Mirsad Hadzikadic, College of Information Technology, UNC Charlotte, USA

Joachim Hammer, University of Bremen, Germany

Alexander Hinneburg, Martin-Luther-University Halle/Wittenberg, Germany

Tu Bao Ho, Japan Advanced Institute of Science and Technology, Japan

Jaakko Hollmen, Helsinki University of Technology, Finland

Se June Hong, IBM – The Data Analytics Research Project, USA

Andreas Hotho, University of Kassel, Kassel, Germany

Matthias Jarke, Fraunhofer Institute for Applied Information Technology FIT,  
Germany

Manfred Jeusfeld, Tilburg University, Netherlands

Hiroyuki Kawano, Nanzan University, Japan

Masaru Kitsuregawa, University of Tokyo, Japan  
Jens Lechtenboerger, University of Muenster, Germany  
Yue-Shi Lee, Ming-Chuan University, Taiwan  
Wolfgang Lehner, Dresden University of Technology, Germany  
Tok Wang Ling, National University of Singapore, Singapore  
Beate List, Vienna University of Technology, Austria  
Yannis Manolopoulos, Aristotle University, Greece  
Mukesh Mohania, IBM India Research Lab, India  
Wee Keong Ng, Nanyang Technological University, Singapore  
Tho Manh Nguyen, Vienna University of Technology, Austria  
Dimitris Papadias, Hong Kong University of Science and Technology, Hong Kong, China  
Torben Bach Pedersen, Aalborg University, Denmark  
Jian Pei, The State University of New York, USA  
Jaakko Peltonen, University of Helsinki, Finland  
Clara Pizzuti, Università della Calabria, Italy  
David Powers, The Flinders University of South Australia, Australia  
Mirek Riedewald, Cornell University, USA  
Stefano Rizzi, University of Bologna, Italy  
Domenico Saccà, Università Della Calabria, Italy  
Monica Scannapieco, University of Rome, Italy  
Josef Schiefer, Vienna University of Technology, Austria  
Markus Schneider, University of Florida, USA  
Michael Schrefl, Johannes Kepler University Linz, Austria  
Timos Sellis, National Technical University of Athens, Greece  
Manuel Serrano, University of Castilla La Mancha, Spain  
Alkis Simitsis, National Technical University of Athens, Greece  
Il-Yeol Song, Drexel University, Philadelphia, USA  
Kian-Lee Tan, National University of Singapore, Singapore  
David Taniar, Monash University, Australia  
Dimitri Theodoratos, New Jersey Institute of Technology, USA  
Riccardo Torlone, Università Roma Tre, Italy  
Panos Vassiliadis, University of Ioannina, Greece  
Roland Wagner, Johannes Kepler University of Linz, Austria  
Werner Winiwarter, University of Vienna, Austria  
Marek Wojciechowski, Poznan University of Technology, Poland  
Wolfram Wöß, University of Linz, Austria  
Xindong Wu, University of Vermont, USA  
Show-Jane Yen, Fu Jen Catholic University, Taiwan  
Mohammed J. Zaki, Rensselaer Polytechnic Institute, USA  
Long Zhang, IBM China Research Laboratory, China  
Shichao Zhang, Sydney University of Technology, Australia

## External Reviewers

Spiridon Bakiras  
Stephan Bloehdorn  
Wim den Boer  
Mario Cannataro  
Tania Cerquitelli  
Eugenio Cesario  
Ellery H.Y. Chen  
Yaohua Chen  
Silvia Chiusano  
Namyoun Choi  
Shian-en Chung  
Pedro Furtado  
Paolo Garza  
Gyozo Gidofalvi  
Daniela Grigori  
Hannes Heikinheimo  
John Horner  
Juliana Hsieh  
Min-Chi Hsieh  
Tok Wee Hyong  
Mika Ilvesmäki  
Yannis Karydis  
Dimitris Katsaros  
Zoubida Kedad  
Habil Hagen Langer  
Rily Tung-Ying Lee  
Erik C. Leertouwer  
Xiang Lian  
Yen-Kuang Lu

Giuseppe Manco  
Alex Markowetz  
Sergio Mascetti  
Massiliano Mazzeo  
Diego Milano  
Deborah E.G. Moolenaar  
Kyriakos Mouratidis  
Alex Nanopoulos  
Apostolos Papadopoulos  
Byung-Kwon Park  
Giovanni Pilato  
Pawel Placek  
Daniele Riboni  
Min Song  
Steffi Soo  
Andrea Tagarelli  
Nikolaj Tatti  
Manolis Terrovitis  
Christian Thomsen  
Igor Timko  
Haorianto Cokrowijoyo Tjioe  
Aris Tsois  
Antti Ukkonen  
Marco Vieira  
Wugang Xu  
Yin Yang  
Hui Zhao  
Yan Zhao



# Table of Contents

## Data Warehouse I

A Tree Comparison Approach to Detect Changes in Data Warehouse Structures <i>Johann Eder, Christian Koncilia, Karl Wiggisser</i> .....	1
Extending the UML for Designing Association Rule Mining Models for Data Warehouses <i>José Jacobo Zubcoff, Juan Trujillo</i> .....	11
Event-Feeded Dimension Solution <i>Tho Manh Nguyen, Jaromir Nemeč, Martin Windisch</i> .....	22
XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses <i>Byung-Kwon Park, Hyoil Han, Il-Yeol Song</i> .....	32

## Data Warehouse II

Graph-Based Modeling of ETL Activities with Multi-level Transformations and Updates <i>Alkis Simitsis, Panos Vassiliadis, Manolis Terrovitis, Spiros Skiadopoulos</i> .....	43
Extending UML 2 Activity Diagrams with Business Intelligence Objects <i>Veronika Stefanov, Beate List, Birgit Korherr</i> .....	53
Automatic Selection of Bitmap Join Indexes in Data Warehouses <i>Kamel Aouiche, Jérôme Darmont, Omar Boussaïd, Fadila Bentayeb</i> .....	64

## Evaluating Data Warehouses and Tools

A Survey of Open Source Tools for Business Intelligence <i>Christian Thomsen, Torben Bach Pedersen</i> .....	74
DWEB: A Data Warehouse Engineering Benchmark <i>Jérôme Darmont, Fadila Bentayeb, Omar Boussaïd</i> .....	85

A Set of Quality Indicators and Their Corresponding Metrics for Conceptual Models of Data Warehouses  
*Gema Berenguer, Rafael Romero, Juan Trujillo, Manuel Serrano, Mario Piattini* . . . . . 95

Design and Development of a Tool for Integrating Heterogeneous Data Warehouses  
*Riccardo Torlone, Ivan Panella* . . . . . 105

**Schema Transformations**

An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse  
*Ladjel Bellatreche, Kamel Boukhalfa* . . . . . 115

Using Schema Transformation Pathways for Incremental View Maintenance  
*Hao Fan* . . . . . 126

Data Mapper: An Operator for Expressing One-to-Many Data Transformations  
*Paulo Carreira, Helena Galhardas, João Pereira, Antónia Lopes* . . . . . 136

**Materialized Views**

Parallel Consistency Maintenance of Materialized Views Using Referential Integrity Constraints in Data Warehouses  
*Jinho Kim, Byung-Suk Lee, Yang-Sae Moon, Soo-Ho Ok, Wookey Lee* . . . . . 146

Selective View Materialization in a Spatial Data Warehouse  
*Songmei Yu, Vijayalakshmi Atluri, Nabil Adam* . . . . . 157

PMC: Select Materialized Cells in Data Cubes  
*Hongsong Li, Houkuan Huang, Shijin Liu* . . . . . 168

**Aggregates**

Progressive Ranking of Range Aggregates  
*Hua-Gang Li, Hailing Yu, Divyakant Agrawal, Amr El Abbadi* . . . . . 179

On Efficient Storing and Processing of Long Aggregate Lists  
*Marcin Gorawski, Rafal Malczok* . . . . . 190

## Data Warehouse Queries and Database Processing Issues

Ad Hoc Star Join Query Processing in Cluster Architectures <i>Josep Aguilar-Saborit, Victor Muntés-Mulero, Calisto Zuzarte, Josep-L. Larriba-Pey</i> .....	200
A Precise Blocking Method for Record Linkage <i>Patrick Lehti, Peter Fankhauser</i> .....	210
Flexible Query Answering in Data Cubes <i>Sami Naouali, Rokia Missaoui</i> .....	221
An Extendible Array Based Implementation of Relational Tables for Multi Dimensional Databases <i>K.M. Azharul Hasan, Masayuki Kuroda, Naoki Azuma, Tatsuo Tsuji, Ken Higuchi</i> .....	233

## Data Mining Algorithms and Techniques

Nearest Neighbor Search on Vertically Partitioned High-Dimensional Data <i>Evangelos Dellis, Bernhard Seeger, Akrivi Vlachou</i> .....	243
A Machine Learning Approach to Identifying Database Sessions Using Unlabeled Data <i>Qingsong Yao, Xiangji Huang, Aijun An</i> .....	254
Hybrid System of Case-Based Reasoning and Neural Network for Symbolic Features <i>Kwang Hyuk Im, Tae Hyun Kim, Sang Chan Park</i> .....	265

## Data Mining

Spatio-temporal Rule Mining: Issues and Techniques <i>Győző Gidófalvi, Torben Bach Pedersen</i> .....	275
Hybrid Approach to Web Content Outlier Mining Without Query Vector <i>Malik Agyemang, Ken Barker, Reda Alhadj</i> .....	285
Incremental Data Mining Using Concurrent Online Refresh of Materialized Data Mining Views <i>Mikołaj Morzy, Tadeusz Morzy, Marek Wojciechowski, Maciej Zakrzewicz</i> .....	295

A Decremental Algorithm for Maintaining Frequent Itemsets in  
Dynamic Databases  
*Shichao Zhang, Xindong Wu, Jilian Zhang, Chengqi Zhang* . . . . . 305

## Association Rules

Discovering Richer Temporal Association Rules from Interval-Based  
Data  
*Edi Winarko, John F. Roddick* . . . . . 315

Semantic Query Expansion Combining Association Rules with  
Ontologies and Information Retrieval Techniques  
*Min Song, Il-Yeol Song, Xiaohua Hu, Robert Allen* . . . . . 326

Maintenance of Generalized Association Rules Under Transaction  
Update and Taxonomy Evolution  
*Ming-Cheng Tseng, Wen-Yang Lin, Rong Jeng* . . . . . 336

Prince: An Algorithm for Generating Rule Bases Without Closure  
Computations  
*Tarek Hamrouni, Sadok Ben Yahia, Yahya Slimani* . . . . . 346

## Text Processing and Classification

Efficient Compression of Text Attributes of Data Warehouse  
Dimensions  
*Jorge Vieira, Jorge Bernardino, Henrique Madeira* . . . . . 356

Effectiveness of Document Representation for Classification  
*Ding-Yi Chen, Xue Li, Zhao Yang Dong, Xia Chen* . . . . . 368

2-PS Based Associative Text Classification  
*Tieyun Qian, Yuanzhen Wang, Hao Long, Jianlin Feng* . . . . . 378

## Miscellaneous Applications

Intrusion Detection via Analysis and Modelling of User  
Commands  
*Matthew Gebski, Raymond K. Wong* . . . . . 388

Dynamic Schema Navigation Using Formal Concept Analysis  
*Jon Ducrou, Bastian Wormuth, Peter Eklund* . . . . . 398

## Security and Privacy Issues

FMC: An Approach for Privacy Preserving OLAP <i>Ming Hua, Shouzhi Zhang, Wei Wang, Haofeng Zhou, Baile Shi</i> . . . . .	408
Information Driven Evaluation of Data Hiding Algorithms <i>Elisa Bertino, Igor Nai Fovino</i> . . . . .	418

## Patterns

Essential Patterns: A Perfect Cover of Frequent Patterns <i>Alain Casali, Rosine Cicchetti, Lotfi Lakhal</i> . . . . .	428
Processing Sequential Patterns in Relational Databases <i>Xuequn Shang, Kai-Uwe Sattler</i> . . . . .	438
Optimizing a Sequence of Frequent Pattern Queries <i>Mikołaj Morzy, Marek Wojciechowski, Maciej Zakrzewicz</i> . . . . .	448
A General Effective Framework for Monotony and Tough Constraint Based Sequential Pattern Mining <i>Enhong Chen, Tongshu Li, Phillip C-y Sheu</i> . . . . .	458

## Cluster and Classification I

Hiding Classification Rules for Data Sharing with Privacy Preservation <i>Juggapong Natwichai, Xue Li, Maria Orlowska</i> . . . . .	468
Clustering-Based Histograms for Multi-dimensional Data <i>Filippo Furfaro, Giuseppe M. Mazzeo, Cristina Sirangelo</i> . . . . .	478
Weighted K-Means for Density-Biased Clustering <i>Kittisak Kerdprasop, Nittaya Kerdprasop, Pairote Sattayatham</i> . . . . .	488

## Cluster and Classification II

A New Approach for Cluster Detection for Large Datasets with High Dimensionality <i>Matthew Gebski, Raymond K. Wong</i> . . . . .	498
Gene Expression Biclustering Using Random Walk Strategies <i>Fabrizio Angiulli, Clara Pizzuti</i> . . . . .	509

Spectral Kernels for Classification <i>Wenyuan Li, Kok-Leong Ong, Wee-Keong Ng, Aixin Sun</i> . . . . .	520
Data Warehousing and Knowledge Discovery: A Chronological View of Research Challenges <i>Tho Manh Nguyen, A Min Tjoa, Juan Trujillo</i> . . . . .	530
<b>Author Index</b> . . . . .	537

# A Tree Comparison Approach to Detect Changes in Data Warehouse Structures

Johann Eder, Christian Koncilia, and Karl Wiggisser

University of Klagenfurt,  
Dep. of Informatics-Systems  
{eder, koncilia, wiggisser}@isys.uni-klu.ac.at

**Abstract.** We present a technique for discovering and representing changes between versions of data warehouse structures. We select a tree comparison algorithm, adapt it for the particularities of multidimensional data structures and extend it with a module for detection of node renamings. The result of these algorithms are so called editscripts consisting of transformation operations which, when executed in sequence, transform the earlier version to the later, and thus show the relationships between the elements of different versions of data warehouse structures. This procedure helps data warehouse administrators to register changes. We describe a prototypical implementation of the concept which imports multidimensional structures from Hyperion Essbase data warehouses, compares these versions and generates a list of differences.

## 1 Introduction and Motivation

Data warehouses provide sophisticated features for aggregating, analyzing, and comparing data to support decision making in companies. The most popular architecture for data warehouses are multidimensional data cubes, where transaction data (called cells, fact data or measures) are described in terms of master data (also called dimension members). Usually, dimension members are hierarchically organized in dimensions, e.g., *university*  $\leftarrow$  *faculty*  $\leftarrow$  *department*. where  $B \leftarrow A$  means that  $A$  rolls-up to  $B$ .

As most data warehouses typically comprise a time dimension, available data warehouse systems are able to deal with changing measures, e.g., changing margin or sales. They are however not able to deal with modifications in dimensions, e.g., if a new faculty or department is established, or a faculty is split into two, or departments are joined.

In [1] we presented the COMET approach, a temporal data warehouse meta-model, which allows to represent not only changes of transaction data, but also of schema, and structure data. The COMET model can then be used as basis of OLAP tools which are aware of structural changes and permit correct query results spanning multiple periods and thus different versions of dimension data.

Temporal data warehouses, however, need a representation of changes which took place between succeeding structure versions of the dimension data. Typically, a change log is not available, but the data warehouse administrator has

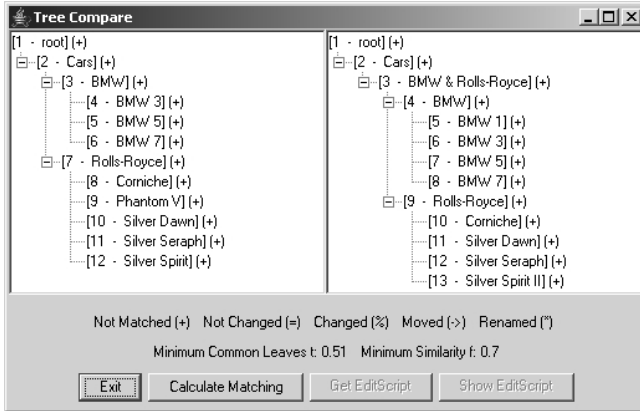


Fig. 1. Our running Example

to create and maintain a mapping between snapshots of the dimension structure. The contribution of this paper is to assist data warehouse administrators in detecting and describing these structural changes, i.e. insertion, deletion and rearrangement of dimension members. Besides these structural changes in tree structured dimensions there are also semantical changes (e.g. merging or splitting of dimension members) which cannot be discovered by looking at dimension data alone. For these semantical changes we have developed a change detection procedure based on data mining techniques [2]. Since this method analyzes outliers in the cell values, the computational costs are rather high. Therefore, it is desirable to find structural changes first by mere structural comparisons with more efficient algorithms.

For detecting structural changes, we present a novel comparison algorithm for multidimensional data by adopting and extending an existing tree comparison algorithm to the particularities of data warehouses. One of the main advantages of our approach is that beside detecting changes like insert, delete and update of an element on the instance level, it also supports the detection of key modifications (e.g. renamings of departments).

Such an approach can only be heuristic in nature and can never be complete, as we will explain below. Therefore, this method is intended to support the administrator but not to fully automate the task. Since dimension members can be numerous (e.g. product catalog), the productivity gain will be significant.

Throughout the rest of this paper, we will use the following running example. Consider a car dealer who wants to keep track of her/his sales. For this, she/he implements a data warehouse with several dimensions. For sake of simplicity, we take a closer look at only one of these dimensions, namely the Cars dimension.

The left tree in Fig. 1 depicts the original version of this dimension. As can be seen, this dealer sells two different brands: *BMW* and *Rolls-Royce*. Each brand consists of several different car types. For instance, *Corniche*, *Phantom V* and *Silver Dawn* are different Rolls-Royce cars. The right tree in Fig. 1 shows the



subsequent version of this dimension. As can be seen, different modifications have been made: first of all, both brands united and are now known as *BMW & Rolls-Royce*. A new car was introduced, namely *BMW 1*. Another car, *Phantom V*, is no longer part of the product portfolio. *Silver Spirit* has been renamed and is now known as *Silver Spirit II*. Moreover, for all Rolls-Royce cars power is no longer given in kW but in horsepower.

## 2 Related Work

Our approach builds on the techniques developed in two different research areas, namely temporal data warehousing and tree comparison algorithms.

During the last years different temporal data warehouse approaches have been published. [3,4,5,6,7,8] are just a few of them. They differ in different aspects. For instance some of them support only changes on the schema level (e.g. [7]) or on the instance level (e.g. [3] and [4]), some support changes on both, the schema and the instance level (e.g. [8]).

There are several tree comparison algorithms. For instance, Zhang and Shasha [9] worked on ordered trees. The Stanford DB Group proposed algorithms for ordered [10] and unordered [11] trees. Nowadays as XML is very popular many algorithms for comparing XML documents, which are also trees, have been defined, for instance the approach of Cobena, Abiteboul and Marian [12] or the approach of Wang, DeWitt and Cai [13].

## 3 Comparison of Data Warehouse Structures

### 3.1 The Data Structure

From our running example it is easy to see, that a DWH structure can be represented by a tree. To define our data structure formally, we introduce a tree  $\mathbb{T} = (\mathbb{V}, \mathbb{E})$ , where  $\mathbb{V} = \{m_1, \dots, m_m\}$  is a set of nodes and  $\mathbb{E} = \{e_1, \dots, e_n\}$  is a set of edges. A node is representing a dimension member of a DWH cube, therefore node  $m_i$  is defined as triple  $m_i = \langle id, label, value \rangle$ , where *id* is a unique identifier for each node, *label* is the dimension member's name and *value* is an object containing all other characteristics – i.e. formula, alias, description, consolidation, ... – of the dimension member. The id may stem from the data source or may be generated during the process of building the tree. An edge  $e_i$  is a hierarchical relation between two members and therefore defined as tuple  $e_i = \langle m_j, m_k \rangle$ , meaning that  $m_j$  is the parent of  $m_k$ .  $\mathbb{E}^+$  is the transitive closure of  $\mathbb{E}$  and therefore holding all ancestor relations.

Depending on the underlying DWH system, the structure can either be seen as *ordered tree* or *unordered tree*. In an ordered tree, members have a designated order within their parents, whereas in an unordered tree they don't. To be able to identify the nodes in the tree and to map them to the underlying DWH-system, we define labels to be unique in an unordered tree. As we always can identify a node through its parent and position within the parent in ordered trees, labels don't have to be unique in such a tree.

### 3.2 Transformations and Operations

We compare two versions of a DWH structure. Therefore we represent each version by a tree. Between this two versions a sequence of tree transformations occurred. Our goal is to find this transformations.

We identified five possible operations on the member level. Hereafter  $t_1 : \mathbb{T} = (\mathbb{V}, \mathbb{E})$  references the old version of a tree and  $t_2 : \mathbb{T} = (\mathbb{V}', \mathbb{E}')$  references the new version. Hence  $\mathbb{V}, \mathbb{E}$  and  $\mathbb{V}', \mathbb{E}'$  are the sets of nodes and edges before and after the transformation respectively.

1. **DELETE** ( $\text{DEL}(m_i)$ ): The dimension member  $m_i$  is deleted from the DWH structure. A node can only be deleted if it does not have children.
  - (a) Precondition:  $m_i \in \mathbb{V}, \exists e_j = \langle -, m_i \rangle \in \mathbb{E}, \nexists e_k = \langle m_i, - \rangle \in \mathbb{E}$
  - (b) Operation:  $\mathbb{V}' = \mathbb{V} \setminus \{m_i\}, \mathbb{E}' = \mathbb{E} \setminus \{e_j\}$
  - (c) Postcondition:  $m_i \notin \mathbb{V}', \nexists e = \langle -, m_i \rangle \in \mathbb{E}'$
2. **INSERT** ( $\text{INS}((m_i, l, v), m_j, m_k)$ ): The member  $m_i$  with label  $l$  and value  $v$  is inserted as child of node  $m_j$  directly after node  $m_k$ . If  $m_k$  is *NULL*,  $m_i$  becomes the first child of  $m_j$ . For unordered trees  $m_k$  may always be *NULL*.
  - (a) Precondition:  $m_i \notin \mathbb{V}, m_j \in \mathbb{V}, m_k = \text{NULL} \vee (m_k \in \mathbb{V} \wedge (m_j, m_k) \in \mathbb{E})$
  - (b) Operation:  $\mathbb{V}' = \mathbb{V} \cup \{m_i\}, \mathbb{E}' = \mathbb{E} \cup \{\langle m_j, m_i \rangle\}$
  - (c) Postcondition  $m_i \in \mathbb{V}', \langle m_j, m_i \rangle \in \mathbb{E}'$
3. **MOVE** ( $\text{MOV}(m_i, m_j, m_k)$ ): The dimension member  $m_i$  is moved to the parent  $m_j$  or is moved within its parent  $m_j$  to be directly after  $m_k$ . If  $m_k$  is *NULL*,  $m_i$  becomes the first child of  $m_j$ . For unordered trees  $m_k$  may be *NULL*.
  - (a) Precondition:  $m_i \in \mathbb{V}, m_j \in \mathbb{V}, m_k = \text{NULL} \vee (m_k \in \mathbb{V} \wedge (m_j, m_k) \in \mathbb{E}), \langle m_i, m_j \rangle \notin \mathbb{E}^+$
  - (b) Operation:  $\mathbb{V}' = \mathbb{V}, \mathbb{E}' = (\mathbb{E} \setminus \langle -, m_i \rangle) \cup \{\langle m_j, m_i \rangle\}$
  - (c) Postcondition  $\langle m_j, m_i \rangle \in \mathbb{E}'$
4. **UPDATE** ( $\text{UPD}(m_i, \text{value})$ ): The characteristics of a dimension member  $m_i$  – i.e. the node's value – is changed to *value*.
  - (a) Precondition:  $m_i = \langle id, label, - \rangle \in \mathbb{V}$
  - (b) Operation:  $\mathbb{E}' = \mathbb{E}, m'_i = \langle id, label, value \rangle, \mathbb{V}' = (\mathbb{V} \setminus \{m_i\}) \cup \{m'_i\}$
  - (c) Postcondition:  $m'_i \in \mathbb{V}' = \langle id, label, value \rangle$
5. **RENAME** ( $\text{REN}(m_i, \text{label})$ ): The name of a dimension member  $m_i$  – i.e. the node's label – is changed to *label*.
  - (a) Precondition:  $m_i = \langle id, -, value \rangle \in \mathbb{V}$
  - (b) Operation:  $\mathbb{E}' = \mathbb{E}, m'_i = \langle id, label, value \rangle, \mathbb{V}' = (\mathbb{V} \setminus \{m_i\}) \cup \{m'_i\}$
  - (c) Postcondition:  $m'_i \in \mathbb{V}' = \langle id, label, value \rangle$

One may argue that **MOVE** is not a basic operation, as it may be composed using **DELETE** and **INSERT**. This is true for most cases, but in our context we want to identify relations between the old and the new version of a member. This enables us to define what we called *transformation functions* to transform cell data between different versions [1].

We distinguish between **UPDATE** and **RENAME** because in many commercial systems, e.g. Hyperion Essbase, the member's name is a key and we want to distinguish between value changes and key changes of a member.

### 3.3 Comparison of Dimension Members

To compare different versions of dimension members, we define a  $compare(x, y)$  function that takes into account the various characteristics of a dimension member. The  $compare(x, y)$  function takes two value-objects as defined above as parameters and returns a value in the range of  $[0, 1]$  as *degree of similarity*. Characteristics to be compared may for instance include formulae, user defined attributes (UDAs), aliases, comments, shared member relations, consolidation function and other DWH system specific attributes.

Some of these attributes are more distinguishing than others. So if for example two nodes have exactly the same formula and three out of four UDAs in common but a different consolidation function, it is rather likely that they represent the same element, so  $compare(x, y)$  gives a value near to 1. On the other hand, if the consolidation function is the same, but one is a shared member and the other is not, it may be quite unlikely that these nodes represent the same element, hence  $compare(x, y)$  results in a value near to 0. We define these weighting factors to be *parametrizeable*. Hence the user may decide what “similar” exactly means in her/his situation.

## 4 Treecomparison and Extensions

As tree comparison is a well explored area, there was no need to develop a new comparison algorithm for trees. Instead we evaluated different existing methods in order to find the ideal base for our approach.

In [10] Chawathe et al. present an algorithm for comparing two versions of a tree. The result of the algorithm is an *editscript* consisting of tree transformations which transforms  $t_1$  into  $t_2$ . The time complexity of this method is  $O(nd + d^2)$ , where  $n$  is the *number of leaves* in the tree and  $d$  is a measure for the *difference of the two versions*. As we assume that there are only a few changes between two versions, we can say that  $d \ll n$ . We chose Chawathe et al.’s algorithm as base for our work because of a couple of reasons: its support of the essential MOVE operation, its low time complexity, its ready-to-use editscript as representation for changes, and it can be used for ordered and unordered trees.

In this section we will introduce Chawathe et al.’s treecomparison algorithm and our extensions to make it applicable for our domain.

### 4.1 Treecomparison in Detail

Chawathe et al.’s algorithm is defined on ordered trees but is applicable on unordered trees as well. Each node has a label and a value which are obtained from the data source. Furthermore, each node has a unique identifier which can either stem from data source or be generated during data extraction. The id may not identify nodes over different versions, so nodes representing the same elements in different versions may have different ids and vice versa.

The result of this procedure is a so called *editscript* which transforms  $t_1$  into  $t_2$ . This editscript is a sequence consisting of four atomic operations: INSERT,

UPDATE, MOVE and DELETE. As the matching of nodes between versions relies on node labels, change of node labels (RENAME) is not supported.

Chawathe et al. define their INSERT and MOVE operations index based, meaning that to determine the position where to insert a node, they use the index within the children of a node. To adapt their approach to our predecessor based data structure we modified the  $FindPos(x)$  function to return the predecessor for the node, or  $NULL$ , if there is no predecessor, instead of the index.

The algorithm compares two tree versions:  $t_1 : \mathbb{T}$  and  $t_2 : \mathbb{T}$ . For simplicity reasons we write  $x \in t$  meaning that node  $x$  is an element of the node set  $\mathbb{V}$  of tree  $t$ . The comparison algorithm needs the two tree versions and a matching set as input. The matching set  $\mathbb{M}$  is a set of pairs  $(a, b)$  with  $a \in t_1$  and  $b \in t_2$ , where  $a$  and  $b$  represent the same member in the two versions.

As the node ids may not identify elements over versions other *matching criteria* for nodes have been defined. Before we can give a definition of these criteria, we have to introduce some terms:  $l(x)$  and  $v(x)$  give the label and value of node  $x$  respectively. The function  $common(x, y)$  gives  $\{(v, w) \in \mathbb{M} \mid v \text{ is leaf descendant of } x \text{ and } w \text{ is leaf descendant of } y\}$  the so called *Common Leaves* of  $x$  and  $y$ . Finally  $|x|$  is the number of leaf descendants of node  $x$ . We also slightly modified the original matching criteria for leaf nodes for sake of simplicity. So two nodes  $x$  and  $y$  are seen as representing the same element iff one of the following conditions holds:

1.  $x$  and  $y$  are *leaves*,  $l(x) = l(y)$ , and  $compare(v(x), v(y)) \geq f$ , where  $f$  is a user defined threshold in the range  $[0, 1]$ .  $f$  is called *Minimum Similarity*.
2.  $x$  and  $y$  are *inner nodes* and  $l(x) = l(y)$  and  $\frac{|common(x, y)|}{\max(|x|, |y|)} \geq t$  for  $\frac{1}{2} < t \leq 1$ . We call  $t$  the *Minimum Common Leaves*.  $t$  is defined by the user.

It can easily be seen that in addition to their label leaf nodes are compared using their values, but inner nodes are only compared using their descendants. For sure changes in the values of inner nodes will be detected later on in the update phase. Zhang [14] proposes a similar matching constraint for inner nodes. As labels have to be equal for allowing matches, renamings of nodes – i.e. changes of labels – cannot be detected in the approach of Chawathe et al.

After the matching set  $\mathbb{M}$  is calculated the edit script can be generated. This happens in five phases.

1. Update Phase: For all pairs of nodes  $(x, y) \in \mathbb{M}$  where  $v(x) \neq v(y)$  an update operation is generated.
2. Align Phase: For all misaligned nodes, i.e. if the order is different in the two versions, appropriate move operations are generated.
3. Insert Phase: For all unmatched nodes  $x \in t_2$  an insert is generated.
4. Move Phase: For all pairs of nodes  $(x, y) \in \mathbb{M}$  such that  $(p(x), p(y)) \notin \mathbb{M}$  ( $p(x)$  is the parent of  $x$ ) a move to the correct parent is generated.
5. Delete Phase: For all unmatched nodes  $x \in t_1$  a delete operation is generated.

## 4.2 Detecting Renaming of Nodes

With the plain algorithm described above changes of labels – which are renamings of dimension members in our case – cannot be detected. Hence, our approach extends Chawathe et al.’s approach in order to *detect label changes*.

In the original algorithm a renaming will always result in two operations, one DELETE and one INSERT. But this does not represent the real semantics. We want the different versions of members to be connected over structure versions. Therefore, we introduce the new operation RENAME, denoted as  $\text{REN}(x, l)$  where  $x$  is the node to be renamed and  $l$  is its new label.

The renaming detection takes place after the matching set calculation but before generating the editscript. All nodes  $x \in t_1$  and  $y \in t_2$  which are not part of a matching may have been renamed. They are added to the sets *OldNames*  $\mathbb{O}$  and *NewNames*  $\mathbb{N}$ , respectively. We define  $\mathbb{O} = \{x \in t_1 \mid \#(x, b) \in \mathbb{M}\}$  and  $\mathbb{N} = \{y \in t_2 \mid \#(a, y) \in \mathbb{M}\}$ . If one of the sets is empty, no renaming is possible.

For reduction of complexity we only consider *renamings within the same parent* to be detected. So parents of possibly renamed nodes have to match. But as the parents may also be renamed and therefore no match is possible yet, this rule may foreclose detecting many renamings. Hence we also consider possibly renamed parents as matched parents. For this purpose we define a set *PossibleRenamings*  $\mathbb{P}$  as follows:  $\mathbb{P} = \{(a, b) \mid a \in \mathbb{O} \wedge b \in \mathbb{N} \bullet (p(a), p(b)) \in \mathbb{M} \vee (p(a), p(b)) \in \mathbb{P}\}$ . So  $\mathbb{P}$  is the set of all pairs of nodes from  $\mathbb{O}$  and  $\mathbb{N}$  where their parents either match or are possibly renamed. One can create  $\mathbb{P}$  during a Top-Down traversal of the trees or by a repeated application of the build rule until the set remains stable.

We also define an *order* on  $\mathbb{P}$  which is important for the appropriate traversal of  $\mathbb{P}$  in the next step. To define this order formally, we introduce the *level* of a member  $x$  ( $\text{lev}(x)$ ) as the height of the subtree rooted at  $x$ . We define the operator “ $<$ ” on pairs  $\in \mathbb{P}$  as follows:  $\forall (a, b), (x, y) \in \mathbb{P} : (a, b) < (x, y) \Leftrightarrow \min(\text{lev}(a), \text{lev}(b)) < \min(\text{lev}(x), \text{lev}(y))$ . The order within leaf node pairs and inner node pairs respectively is irrelevant. So if  $\mathbb{P}$  is traversed following this order, all pairs containing at least one leaf node will be examined before any pair containing only inner nodes.

The similarity check for renamed nodes has in principle the same constraints as mentioned before. So we define *likelyRenamed*( $a, b$ ) which checks if  $(a, b) \in \mathbb{P}$  is a likely renaming, to return *true* iff one of the following conditions holds:

1.  $a$  and  $b$  are leaf nodes and  $\text{compare}(v(a), v(b)) \geq f$  ( $f$  as defined above)
2.  $a$  and  $b$  are inner nodes and  $\frac{|\text{commonRename}(a, b)|}{\max(|a|, |b|)} \geq t$  ( $t$  as defined above)

The function *commonRename*( $x, y$ ) gives  $\{(v, w) \mid v \text{ is leaf descendant of } x \text{ and } w \text{ is leaf descendant of } y, \text{ and } (v, w) \in \mathbb{M} \text{ or } (v, w) \in \mathbb{L}\}$ , so all common leaves plus all common likely renamed leaf nodes.

In an ordered tree within the calculation of *likelyRenamed*( $a, b$ ) one may also take into account the *siblings* of the nodes. So if the predecessors and the successors of  $a$  and  $b$  match, one may increase the degree of similarity a bit, although it must not reach 1, as this would mean identical.

$likelyRenamed(a, b)$  splits  $\mathbb{P}$  into two disjoint sets *LikelyRenamings*  $\mathbb{L}$  and *UnlikelyRenamings*  $\mathbb{U}$ .  $\mathbb{L}$  contains all pairs of nodes which are sufficiently similar to be seen as representing the same element. All other elements of  $\mathbb{P}$  are moved to  $\mathbb{U}$ . As for one node only one real renaming can have happened, the following restriction has to hold for  $\mathbb{L}$ :  $\forall(a, b), (x, y) \in \mathbb{L} \bullet a = x \Leftrightarrow b = y$ . So a node can only appear in at most one pair in  $\mathbb{L}$ . If more than one likely renaming for one node is detected, we define the one with the highest similarity of the involved nodes to go to  $\mathbb{L}$ , all others are moved to  $\mathbb{U}$ . Because of  $\mathbb{P}$ 's order all leaves are handled first. Likely renamed leaves are used in the similarity check of inner nodes, i.e. likely renamed leaves are seen as common leaves. Therefore, it is important to follow  $\mathbb{P}$ 's order during this step, as otherwise renamed inner nodes may not be detected correctly.

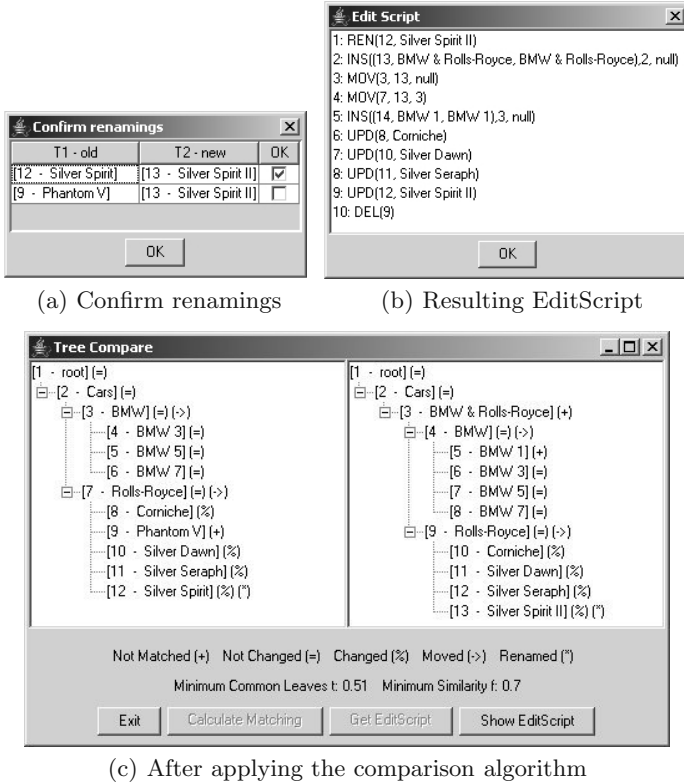
The renaming detection component cannot replace human interaction. It relies on heuristics which may return a wrong result. Therefore, a human user has to acknowledge all detected renamings by checking  $\mathbb{L}$  and  $\mathbb{U}$ . All renamings confirmed by the user are moved into the set *Renamings*  $\mathbb{R}$ . For all pairs  $(a, b) \in \mathbb{R}$  a rename operation  $\mathbf{REN}(a, \mathbf{1}(b))$  is generated and  $(a, b)$  is added to  $\mathbb{M}$ . Furthermore the algorithm may not detect all renamings. For instance, if a node is renamed, its value changed very much and it is moved to another parent, then the renaming will not be detected. For complexity purposes only renamed nodes within the same parent will be considered. One may omit this restriction but this will increase the runtime complexity considerably.

After this verbal description, we now formally describe the steps which are necessary to detect renamings of nodes.

1.  $\mathbb{O} = \{x \in t_1 \mid \nexists(a, b) \in \mathbb{M} \bullet a = x\}$ ,  $\mathbb{N} = \{y \in t_2 \mid \nexists(a, b) \in \mathbb{M} \bullet b = y\}$
2.  $\mathbb{P} = \{(a, b) \mid a \in \mathbb{O}, b \in \mathbb{N} \bullet (p(a), p(b)) \in \mathbb{M} \vee (p(a), p(b)) \in \mathbb{P}\}$
3.  $\forall(a, b) \in \mathbb{P}$  in traversal order  $\bullet$ 
  - (a) if  $likelyRenamed(a, b) = true$ 
    - i.  $\mathbb{P} = \mathbb{P} \setminus \{(a, b)\}$ ,  $\mathbb{L} = \mathbb{L} \cup \{(a, b)\}$
    - ii.  $\forall x \in t_1 \mid (x, b) \in \mathbb{P} \bullet \mathbb{P} = \mathbb{P} \setminus \{(x, b)\}$ ,  $\mathbb{U} = \mathbb{U} \cup \{(x, b)\}$
    - iii.  $\forall y \in t_2 \mid (a, y) \in \mathbb{P} \bullet \mathbb{P} = \mathbb{P} \setminus \{(a, y)\}$ ,  $\mathbb{U} = \mathbb{U} \cup \{(a, y)\}$
  - (b) else  $\mathbb{P} = \mathbb{P} \setminus \{(a, b)\}$ ,  $\mathbb{U} = \mathbb{U} \cup \{(a, b)\}$
4. Let the user acknowledge all real renamings and insert them to  $\mathbb{R}$
5.  $\forall(a, b) \in \mathbb{R} \bullet$ 
  - (a) Generate operation  $\mathbf{REN}(a, \mathbf{1}(b))$
  - (b)  $\mathbb{M} = \mathbb{M} \cup \{(a, b)\}$

## 5 Implementation

We implemented our approach in Java 1.4 under Windows XP. In this prototype the user is able to import and compare two different cubes from the commercial multidimensional database Hyperion Essbase. After importing both cubes from Hyperion Essbase, the prototype presents both versions as trees (see Fig. 1). The left tree represents the old version and right tree the new version of the data warehouse. The user triggers the matching procedure from the interface.



**Fig. 2.** Outcomings of the Algorithm

After the matchings are calculated the system tries to find renamed nodes. Figure 2(a) shows how the user can acknowledge the renamings found by the system. The resulting trees are presented to the user so she/he can evaluate, if the weighting factors are adequate. After the user confirmed the matchings found by the system, the system starts to generate the *editscript*.

The resulting editscript of our running example is shown in Fig. 2(b). Figure 2(c) shows the two trees after the editscript is calculated. Different symbols are used to depict different types of modifications that were detected. “=” means that a corresponding, unchanged dimension member has been found, “->” means that a member has been moved to another position, “%” means that the corresponding member has been changed, e.g., that a user defined attribute has been modified, “+” means that no corresponding member in the other tree could be found. Finally, “\*” means that a member has been renamed.

We also applied our prototype on a larger cube with about 16.400 members. The matching and the editscript generation took in average 0.6 and 1.15 seconds respectively, running on a Pentium IV at 2.4 GHz and 1 GB RAM. Hence we see that this approach may also be applied on large cubes in reasonable time. All changes were recognized correctly.

## 6 Conclusions

For the validity of OLAP queries spanning several periods of data collection it is essential to be aware of the changes in the dimension structure of the warehouse. This means in particular, to have a representation of the changes (and implicit unchanged elements) between different versions of the multidimensional structure.

We adopted and extended a tree comparison algorithm to serve for this purpose. The output of this extended algorithm is an editscript consisting of elementary change operations. We contributed in particular a module for discovering renamings of nodes. Such an editscript facilitates the work of a data warehouse administrator who is in charge of representing structural changes. In a prototype implementation based on Hyperion Essbase we were able to demonstrate the validity of the approach, both the adequacy and validity of the algorithms and their scalability for real data warehouses.

## References

1. Eder, J., Koncilia, C.: Changes of dimension data in temporal data warehouses. In: Proc. of 3rd DaWaK 2001. (2001)
2. Eder, J., Koncilia, C., Mitsche, D.: Automatic Detection of Structural Changes in Data Warehouses. In: Proc. of the 5th DaWaK 2003. (2003)
3. Kimball, R.: Slowly Changing Dimensions, Data Warehouse Architect. DBMS Magazine **9** (1996) URL: <http://www.dbmsmag.com/>.
4. Chamoni, P., Stock, S.: Temporal Structures in Data Warehousing. In: Proc. of the 1st DaWaK 1999. (1999)
5. Yang, J.: Temporal Data Warehousing. PhD thesis, Stanford University (2001)
6. Vaisman, A.: Updates, View Maintenance and Time Management in Multidimensional Databases. PhD thesis, Universidad de Buenos Aires (2001)
7. Blaschka, M.: FIESTA: A Framework for Schema Evolution in Multidimensional Information Systems. PhD thesis, Technische Universität München (2000)
8. Eder, J., Koncilia, C., Morzy, T.: The COMET Metamodel for Temporal Data Warehouses. In: Proc. of the 14th Intl. Conf. on Advanced Information Systems Engineering 2002. (2002)
9. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM journal on computing **18** (1989) 1245–1262
10. Chawathe, S., Rajaraman, A., Garcia-Molina, H., Widom, J.: Change detection in hierarchically structured information. In: Proc. of the 1996 ACM SIGMOD. (1996)
11. Chawathe, S., Garcia-Molina, H.: Meaningful change detection in structured data. In: Proc. of the 1997 ACM SIGMOD. (1997)
12. Cobena, G., Abiteboul, S., Marian, A.: Detecting changes in XML documents. In: Proc. of the 18th Intl. Conf. on Data Engineering. (2002)
13. Wang, Y., DeWitt, D., Cai, J.Y.: X-diff: An effective change detection algorithm for XML documents. In: Proc. of the 19th Intl. Conf. on Data Engineering. (2003)
14. Zhang, L.: On matching nodes between trees. Tech. Rep. 2003–67, HP Labs (2003)



# Extending the UML for Designing Association Rule Mining Models for Data Warehouses

José Jacobo Zubcoff<sup>1</sup> and Juan Trujillo<sup>2</sup>

<sup>1</sup>Departamento de Ciencias del Mar y Biología Aplicada. Universidad de Alicante (Spain)  
Jose.Zubcoff@ua.es

<sup>2</sup>Departamento de Lenguajes y Sistemas Informáticos. Universidad de Alicante (Spain)  
jtrujillo@dlsi.ua.es

**Abstract.** Association rules (AR) are one of the most popular data mining techniques in searching databases for frequently occurring patterns. In this paper, we present a novel approach to accomplish the conceptual design of data warehouses together with data mining association rules, allowing us to implement the association rules defined in the conceptual modeling phase. The great advantage of our approach is that the association rules are specified from the early stages of a data warehouse project and based on the main final user requirements and data warehouse goals, instead of specifying them on the final database implementation structures such as tables, rows or columns. Finally, to show the benefit of our approach we implement the specified association rules on a commercial data warehouse management server.

**Keywords:** Data Warehouses, UML extension, conceptual modeling, multidimensional modeling, Data Mining, KDD, Association rules.

## 1 Introduction

The discovery of interesting association relationships among huge amount of data is called Association Rule (AR). The aim of the AR is to provide an observation a posteriori of the most common links between data. As an example, a customer selects items from those offered by a retailer (market basket). The retailer keeps a registry of every customer transaction and AR lets us know the relationships between the items that customers are purchasing. This is also referred as Market Basket Analysis.

Data warehouses (DW) store historical, cleansed, subject oriented and integrated data extracted from various sources (Fig. 1). Analysts have to collect all business concerned data to implement a data mining (DM) algorithm. Thus, DW is a perfect framework to apply DM. Typically, DW analysts use multidimensional (MD) models to represent the information they manage. In MD models, data is organized into facts (the aim of analysis) and dimensions representing the context where we wish to analyze facts. Therefore, data mining could be modeled in an MD environment.

In order to take advantage of the effort in the modeling process of DW and to enable the potential of algorithms, rules and modeling techniques, to develop in an extended and well-known standard modeling language, we propose to integrate the AR Mining process into DW modeling, extending the UML with a profile.

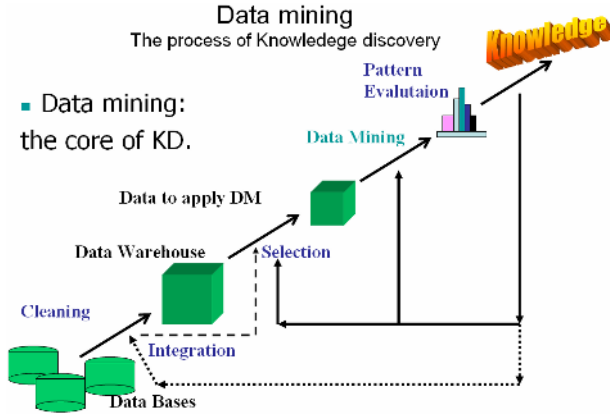


Fig. 1. The process of knowledge discovery

Our proposal is an extension of UML for designing AR Mining models for DW, based on modeling process presented in “Extending the UML for Multidimensional Modeling” [1] which is a UML profile<sup>1</sup> for MD Modeling, that provides the specific mechanisms for conceptual model DW, because it allows us to consider main MD modeling properties and avoids having to learn a new specific notation or language.

The remainder of this paper is structured as follows: Section 2 explains other works that have dealt with association rules, DW and modeling. Section 3 summarizes the conceptual MD modeling on which we are based. Section 4 proposes the new UML extension for designing AR models for MD modeling. Section 5 presents a case study and applies our UML extension for designing AR models for MD modeling, section 6 sketches some further implementation issues. Finally, section 7 presents the main conclusions and introduces immediate and future work.

## 2 Related Work

The mining concept of discovering AR was introduced in [2]. Early proposals used flat files as input data not in a DW environment. Another important contribution was to provide an SQL-like language with a set of primitive operations to support ad-hoc and interactive data mining integrated into databases, specifically extending SQL to support mining operators. DMQL [12], M-SQL [11] and Mine Rule[13] are SQL-like languages. In [3] there is a comparison between the query languages. All efforts to develop a language for DM allow us to integrate the mining process into databases, not in a separated statistic analysis tool, but as a process in a DB framework.

---

<sup>1</sup> A *profile* is a set of improvements that extend an existing UML type of diagram for a different use. These improvements are specified by means of the extendibility mechanisms provided by UML (stereotypes, properties and restrictions) in order to be able to adapt it to a new method or model.

A Pattern Base Management is proposed in [9], this is a system that stores and manages patterns as data is managed by a database management system. It is a novel issue in knowledge discovery, and it has been further modeled in [10], not in DW context but as an isolated system that manages and stores patterns. On the other hand, several data mining approaches have been proposed to run on top of data warehouses. An algorithm for mining in a star schema or multidimensional model was proposed [3], [17]. In [3], [5], [6] mining was applied in relational databases as a sequence of SQL queries, this represents an important advance of the relationship between AR and databases. Another proposal was [18] which represents techniques that improve performance using SQL. At metadata level, in the Data Mining chapter of the Common Warehouse Model (CWM) [7] there is a specification that describes metadata interchange among DW and business intelligence knowledge management. This approach consists of a number of meta-models that represents common warehouse metadata in the major areas of interest to DW and BI, including the mining process. But, in all these approaches the only benefit is that they work with huge data previously cleaned. These works do not take into consideration main final user goals of the corresponding multidimensional model underneath, so they do not use important terms in DW such as facts, dimensions or classification hierarchies.

Therefore, we argue that there is still a missing work that allows us the specification of data mining techniques together with the multidimensional modeling accomplished to design data warehouses. Furthermore, the sooner this is specified (from the early stages of a DW project such as the conceptual modeling phase), the better the specified data mining techniques will be focused on final user needs.

### 3 Object-Oriented Multi-dimensional Modeling

In this section, we outline our approach to DW conceptual modeling, based on the UML [1], [8], [14], [19], specified by means of a UML profile that contains the necessary stereotypes in order to carry out conceptual modeling successfully [15]. The main features of MD modeling are: the many-to-many relationships between the

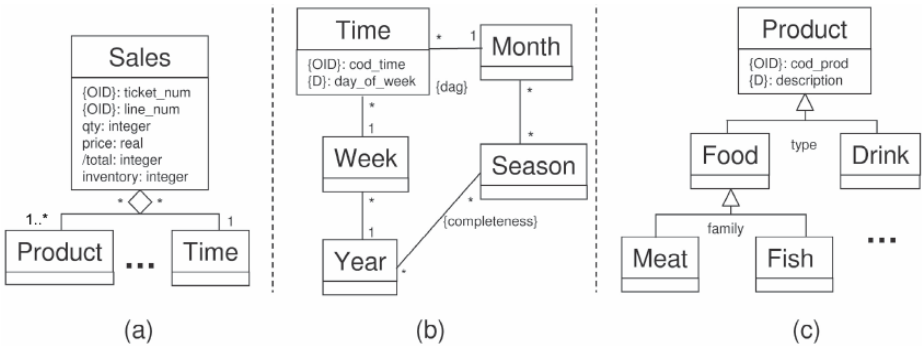


Fig. 2. Multidimensional modeling using the UML

facts and one specific dimension (Fig. 2.a), degenerated dimensions, multiple classification and alternative path hierarchies (Fig. 2.b), the non-strict and complete hierarchies and the categorization of dimensions (Fig. 2.c). In this approach, the structural properties of MD modeling are represented by means of a UML class diagram in which the information is clearly organized into facts and dimensions.

Facts and dimensions are represented by means of fact classes (stereotype Fact) and dimension classes (stereotype Dimension) respectively. Fact classes are defined as composite classes in shared aggregation relationships of dimension classes. The minimum multiplicity in the role of the dimension classes is 1, to indicate that all the facts must always be related to all the dimensions. The relations *many-to-many* between a fact and a specific dimension are specified by means of the multiplicity 1..\* in the role of the corresponding dimension class.

### 4 UML Approach for Association Rule Modeling

Association Rules (AR) look for relationships between items in an entity. This mining process depends on identifying frequent item sets in data, grouped by an entity called Case. Frequent item sets could be used to summarize association between items in the selected attributes. We obtain rules using one or more attributes as input, and one or more attributes to predict. Consequently we have association rules with some input attributes that predicts others (or the same), based on the grouping condition (Case). Consequently, Case from MD point of view could be an –stereotype OID- attribute of a Dimension class, because facts could be aggregated by any dimension or a –stereotype OID- of a Fact class (degenerated Dimension). Input and Predict attributes could be selected from OID attributes (Fact or Dimension) or Fact attributes –stereotype FA- or Dimension attributes –stereotype DA- and rules could have several Input and/or Predict attributes. Case, input and predict are elements of a class called Association Rule Mining Model.

Rules are obtained considering specific settings. Basic settings are minimum support (MinSupp) and minimum confidence (MinConf) that rule must satisfy, max number of records of the item set (MaxItemsetSize) and max number of predicates (MaxNumberOfPredicates) (max number of appearances in the body of the rule). These four settings represent the class Association Rule Mining Settings.

Finally, AR has an antecedent (body) and a consequent (header). The first contains input attribute/s value/s, the last contains predicted attribute/s value/s observed in the grouped item set. AR Mining Results class is built with these four attributes.

Summarizing, AR mining process is represented using three classes: AR Mining Model (ARMM) that is the model, AR Mining Settings (ARMS), and finally AR Mining Results (ARMR) that contains the results of the process, the rules.

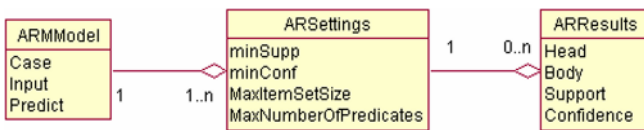


Fig. 3. Association Rule Metamodel

Figure 3 shows the proposed model in which we could see that one AR Mining Model could have several settings, and for each setting we get a set of results.

All conditions take part of Association Rules Settings class (ARSettings) in AR mining model (Fig. 3). Head and Body of the rule are defined as attributes of Association Rule Results class (ARResults), as shown in Fig 3. Each rule obtained has support and confidence, probabilities values that could be represented in percent values, defined as attributes of ARResults class in meta-model.

This proposal allows us to work with attributes from dimensions and facts class of a multidimensional model, to obtain distinct types of association rules: single and multi-dimensional rules, single and multiple predicates rules, hybrid-dimension and inter-dimension association rules, single or multiple-level rules, or any other rules.

There is a “type constraint”: association rule mining applied to DW needs the data under study (Input and Predict) to be discrete. Then quantitative continuous data must be categorized. This is a precondition that introduces a constraint in the model.

Domain expert can provide some additional constraints on the rule pattern to be mined, so that generated rules are of interest to the user and more specific and useful. User could restrict to generate rules about a concrete case of an attribute using OCL.

According to [4], an extension to the UML begins with a brief description and then lists and describes all of the stereotypes, tagged values, and constraints of the extension. In addition to these elements, an extension contains a set of well-formedness rules. These rules are used to determine if a model is semantically consistent with itself. According to this quote, we define our UML extension for AR mining conceptual MD modeling, following the schema composed of these elements: *description, prerequisite extensions, stereotypes / tagged values, well-formedness rules, and comments.*

#### 4.1 Description

This UML extension reuses a set of stereotypes previously defined in [1], and defines a set of tagged values, stereotypes, and constraints, which enables us to create AR mining integrated into MD models. The 18 tagged values we have defined are applied to certain components of the MD modeling, allowing us to represent them in the same model and on the same diagrams that describe the rest of the system. These tagged values will represent the participation in the AR structure of the different elements of the MD modeling (fact class, dimension class, base class, attributes, etc.), allowing us to specify how rules will be obtained depending on this mining model structure information and on the value of attributes of the model. A set of constraints are specified in order to define well-formedness rules. The correct use of our extension is assured by the definition of constraints in both natural language and OCL [16].

#### 4.2 Prerequisite Extensions

This UML profile reuses stereotypes that were previously defined in another UML profile in [1]. This profile provided the needed stereotypes, tagged values, constraints to accomplish the MD modeling properly, allowing us to represent main MD properties of DW's at the conceptual level. To facilitate the comprehension of the UML profile we present and use in this paper, we provide a summary of the specification of these stereotypes in Table 1.

**Table 1.** Stereotype from the UML profile for conceptual MD modeling [19]

Name	Base Class	Description
Fact	Class	Classes of this stereotype represent facts in a MD model
Dimension	Class	Classes of this stereotype represent dimensions in a MD model
Base	Class	Classes of this stereotype represent dimension hierarchy levels in a MD model
OID	Attribute	Represent OID attributes of Facts, Dimension or Base classes in a MD model
Fact-Attributes	Attribute	Attributes of this stereotype represent attributes of Fact classes in a MD model
Descriptor	Attribute	Represent descriptor attributes of Dimension or Base classes in a MD model
Dimension-Attribute	Attribute	Represent attributes of Dimension or Base classes in a MD model
Completeness	Association	Associations of this stereotype represent the completeness of an association between a Dimension class and a Base class or between two Base classes

**Table 2.** Tagged values defined in the Profile

Tagged Values of the Model			
Name	Type	M <sup>1</sup>	Description
Classes	Set(OclType)	1..*	It specifies all classes of model.
Tagged Values of the Class			
Name	Type	M	Description
Case	{ OID }	1	It specifies the case that an instance of this class uses to group by. The default attribute level Case tagged value is OID.
Input	Set({D,OID,DA,FA})	1..*	It specifies a set of Inputs of the AR. The default attribute level Input tagged value is D case exists, otherwise, OID.
Predict	Set({D,OID,DA,FA})	1..*	It specifies a set of Predicts class used to obtain AR. The default attribute level Predict tagged value is D if exists, otherwise, OID.
Tagged Values of the Attribute			
Name	Type	M	Description <sup>2</sup>
Case	{ OID }	1	
Input	Set({D,OID,DA,FA})	1..*	
Predict	Set({D,OID,DA,FA})	1..*	
Tagged Values of the Instance			
Name	Type	M	Description
Case	{ OID }	1	It specifies the case of an instance
Input	Set({D,OID,DA,FA})	1..*	It specifies a set of Inputs for this instance.
Predict	Set({D,OID,DA,FA})	1..*	It specifies the set of Predicates for an instance
Tagged Values of the Constraint			
Name	Type	M	Description
Involved-Classes	Set(OCLType)	0..1	Classes that are involved in a rule, to be enforced in the constraint
Case	{ OID }	1	It specifies the attribute that is the Case of the itemsets.
Input	Set({D,OID,DA,FA})	1..*	It specifies a set of Inputs of the AR
Predict	Set({D,OID,DA,FA})	1..*	It specifies a set of Predicts of the AR
MinSupp	Double	0..1	It specifies the minimum support of the AR
MinConf	Double	0..1	It specifies the minimum confidence of the AR
MISS	Integer	0..1	It specifies the maximum item set size of the rule
MNOP	Integer	0..1	It specifies the maximum number of predicates of the rule

<sup>1</sup> M stands for Multiplicity<sup>2</sup> Due to space constraints, we do not include the descriptions of the tagged values of attributes as they are similar to their counterpart tagged values of classes.

### 4.3 Tagged Values

In this section, we provide the definition of several tagged values for the model, classes, attributes, instances and constraints.

Table 2 shows the tagged values of all elements in this extension. We must set only one Case Tagged Value of the Class for each mining model. The Case tagged value of the class could be defined for a Fact class or for a Dimension class. At attribute level the default Case tagged value must be OID –stereotype OID-. Default tagged values of the attributes of Input and Predict are descriptor attribute -stereotype D- in case they exist, or OID -stereotype OID- otherwise. We must select at least one Input and at least one Predict tagged attribute of the class for each Case tagged value of the class. This means that we could have rules with more than one input and several predict attributes. We could set other than D attribute or OID attribute (from Fact or Dimension class) putting a tagged value of Input or Predict of attribute close to the desired one. If more than one OID exists in a class it is mandatory to set the corresponding tagged value of the attribute. Default value of MinSupp is 0.01 and default value of MinConf is 0.40. Default values of MISS and MNOP are 2000 and 3 respectively. These are attributes of association rule mining settings class.

### 4.4 Well-Formedness Rules

We can identify and specify in both natural language and OCL constraints some well-formedness rules. These rules are grouped in Table 3.

**Table 3.** Well-Formedness constraints

- Correct type of the tagged values:
The Case tagged value should be defined for an OID attribute of a Fact or Dimension class of the model
<b>context</b> Model inv self.classes->forAll(a   a.attributes ->forÄll(c   c.Case) -> notEmpty() implies self.attribute.oclsTypeOf(OID))
- Relationship between Input and Predict tagged values of classes and respective tagged values of its attributes:
If is not defined Input and Predict tagged values for an attribute is settled to D defined for its class.
<b>context</b> Model inv self.classes->forAll(c   c.Input ->forÄll(a   a.attributes.Input)->isEmpty() implies a.attribute.oclsTypeOf(D)=Input ) forAll(c   c.Predict->forÄll(a   a.attributes.Predict)->isEmpty() implies attribute.oclsTypeOf(D)=Predict)
- Categorization of continuous values of an Input and Predict tagged value of attribute
Input and Predict Tagged values must be Type of Integer or must be discrete
<b>context</b> Model inv self.classes->forAll(a   a.attributes ->forÄll(p   p.Predict) -> notEmpty() implies self.attribute.ocType(Integer)) or self.attribute.ocType(Set(String))) self.classes->forAll(a   a.attributes ->forÄll(p   p.Input) -> notEmpty() implies self.attribute.ocType(Integer)) or self.attribute.ocType(Set(String)))

### 4.5 Comments

In addition to the previous constraints, the designer can specify association rules with OCL. If the Input, or Predict values of a class or an attribute depends on the value of an attribute of an instance, it can be expressed as an OCL expression (see Fig. 5).

Normally, AR constraints defined for stereotypes of classes (fact, dimension and base) will be defined by using a UML note attached to the class instance<sup>2</sup> corresponding to the Case of the rule (only one Case for Note). We do not impose any restriction on the content of these notes in order to allow the designer the greatest flexibility, only those imposed by the tagged values definitions.

### 5 A Case Study

The goal of our proposal is to model AR mining in dimensional modeling framework. Typical example is Market Basket analysis modeled as star-schema where Sales are the Fact class and Product, Time and Customer are dimension classes.

To discover rules in a MD model we have to select the case we want to analyze relationships of. To obtain associations of products in a market basket, we have to set a basket as Case with ticket number as key to group by, and select Input and Predict attributes, (*SubCategory.Description*) as Input and Predict from Product dimension to

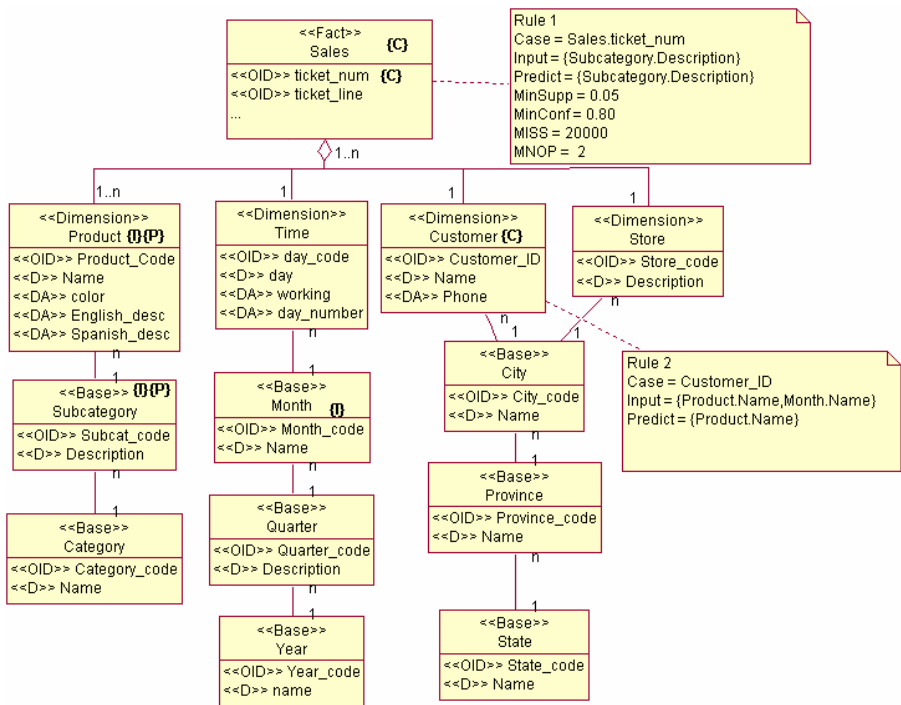


Fig. 4. Example of multidimensional model with AR information and constraints

<sup>2</sup> The connection between a note and the element it applies to is shown by a dashed line without an arrowhead as this is not a dependency [21].



predict which subcategories are related in a market basket. Fig. 4 shows an MD model that includes a fact class (*Sales*), four dimensions (*Product*, *Time*, *Customer* and *Store*) and eight base classes (*Subcategory*, *Category*, *Month*, *Quarter*, *Year*, *City*, *Province* and *State*). *Sales* fact class has two *OID*-stereotype *OID*-: one is *ticket\_num*, which is a degenerated dimension, the other one is *ticket\_line*. Remember that Fact attributes could be used as Case if they are previously categorized.

Adding corresponding tagged values we could obtain different association rules from this model. Two AR constraints have been specified as notes in Fig. 4, each note is attached to the class that contains the corresponding Case tagged value:

Rule 1 uses *Sales.Ticket\_num*-stereotype *OID*- as Case marked as -tagged value *C*- and *Subcategory.Description* -stereotype *D*- marked as -tagged value *I* and *P*- as Input and Predict respectively, the *MinSupp* is 0.05. The *MinConf* is 0.80, *MISS* as a maximum of 20000 frequent itemsets and *MNOP* of 2, are the association rule mining settings. Rules obtained could be “*If helmet and mountain-bike then road bikes (0.52, 0.3)*” which means if a customer buys helmet and mountain-bike also buy road bikes in the 52% of the cases and that this holds in 30% of the transactions.

Rule 2 uses *Customer.ID*-stereotype *OID*- as Case, *Product.Name* -stereotype *D*- and *Month.Name* -stereotype *D*- as Input -tagged value *I* - and *Product.Name* as Predict -tagged value *I* and *P*-. The *MinSupp* is 0.01, *MinConf* is 0.40, *MISS* as a maximum of 2000 frequent itemsets and *MNOP* of 3, by default. Rules obtained could be “*If helmet and September then Road Tire (0.9, 0.04)*”.

## 6 Implementation

The model was implemented in SQL Server 2005 (Beta version). Analysis Services is a component of SQL Server 2005 which allows us to implement DW with the concepts of a MD model. Based on the model used as Case study (Fig. 4) we have created the Fact table *Sales*, the dimensions (*Product*, *Time*, *Customer* and *Store*) and their hierarchies (*Subcategory*, *Category*, *Month*, *Quarter*, *Year*, *City*, *Province* and *State*). Finally we have defined the association Rules. To define the rule 1 in the previous section, we consider Customer dimension as Case, that means that case key is stereotype *OID* of customer, and Subcategory of Product as Input selecting *EnglishProductSubcategory* as key of a nested table as Fig. 5 shown. In SQL Server predict means that will be used as Input and Predict, otherwise use *Predict only*.

Structure		SubCategory by Cust		Parameters:			
			Microsoft_Association_Rules	Parameter	Value	Default	Range
	Dim Customer		Key	MAXIMUM_ITEMSET_COUNT		200000	[1,...]
	Phone		Ignore	MAXIMUM_ITEMSET_SIZE	4	3	[1,500]
	Dim Product 1		Predict	MINIMUM_PROBABILITY		0.4	[0.0,1.0]
	English Product Subcategory		Key	MINIMUM_SUPPORT		0.03	[0.0,...]
	Sales Amount		Ignore				

Fig. 5. Model structure example in Analysis Services

## 7 Conclusions and Future Work

In this paper, we have presented an extension of the UML that allows us to model Association Rules in the conceptual multidimensional modeling of DW. Thus, the defined AR are directly related to the main final user needs and goals of the data warehouse. To achieve this, we have provided the needed stereotypes, tagged values and constraints that allow us to represent AR in multidimensional objects such as facts, dimensions, classification hierarchy levels and so on. To show the benefit of our approach, we have applied our extension to a market basket analysis case study. Finally, we have also shown how the information represented with our approach is implemented on a commercial data base management server with data mining facilities such Microsoft SQL Server 2005 (Beta version). In this way, all AR defined with our approach in the MD modeling at the conceptual level are directly implemented in the DW framework. Our immediate future work is to align our approach with the Model Driven Architecture (MDA) and to extend our profile to represent other data mining techniques rather than just association rules.

## References

- [1] S. Luján-Mora, J. Trujillo and I. Song. *Extending the UML for Multidimensional Modeling*. In Proc. 5<sup>th</sup> International Conference on the UML'02, vol 2460 of LNCS, pages 290-304, Dresden, Germany, September 2002. Springer-Verlag.
- [2] R. Agrawal, T. Imielinski and A. Swami. *Mining Association Rules between Sets of Items in Large Databases*. In Proc. ACM SIGMOD 93, pages 207-216. Washington DC, 1993.
- [3] M. Botta, J. Boulicaut, C. Masson and R. Meo. *A Comparison between Query Languages for the Extraction of Association Rules*. DaWaK 2002: 1-10
- [4] E. Ng, A. Fu, K. Wang. *Mining Association Rules from Stars*. ICDM (IEEE) Maebashi TERRSA, Maebashi City, Japan December 9 - 12, 2002, pages 322-329.
- [5] L. Dehaspe and L. Raedt. *Mining association rules in multiple relations*. In Proc. of the 7th Workshop on ILP, vol. 1297, pages 125-132, Prague, Czech Republic, 1997.
- [6] S. Nestorov, N. Jukić. *Ad-Hoc Association-Rule Mining within the Data Warehouse*. Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
- [7] OMG: *CWM Common Warehouse Metamodel Specification*. <http://www.omg.org>.
- [8] OMG: *UML Unified Modeling Language Specification 1.5*. 2004.
- [9] S. Rizzi et al. *Toward a logical model for patterns*. In Proc. ER Conference, pages 77-90, Chicago, 2003.
- [10] S. Rizzi. *UML-Based Conceptual Modeling of Pattern-Bases*. In Proc. 1st Int. Workshop on "Pattern Representation and Management (PaRMA'04), Crete, Greece, March 2004.
- [11] T. Imielinski, A. Virmani. *MSQL: A Query Language for Database Mining*. Data Mining and Knowledge Discovery, 3. 1999: 373-408
- [12] J. Han, J. Fu, W. Wang, K. Koperski, O. Zaiane. *DMQL: A Data Mining Query Language for Relational Databases*. In SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96), Montreal, Canada, 1996.
- [13] R.Meo, G.Psaila, and S.Geri. *A new SQL-like operator for mining association rules*. In Proceedings of the 2nd Int'l Conference on Very Large Databases, India. September 1998

- [14] J. Trujillo, et al., *Designing Data Warehouses with OO Conceptual Models*. IEEE Computer, special issue on Data Warehouses, 2001(34): p. 66-75.
- [15] M. Gogolla and B. Henderson. *Analysis of UML Stereotypes within the UML Metamodel*. 5th Int Conf. on the UML- The Language and its Applications. 2002. Dresden, Springer,
- [16] J. Warmer and A. Kleppe. *The Object Constraint Language Second Edition. Getting Your Models Ready for MDA*. 2003: Addison Wesley.
- [17] H. Günzel, J. Albrecht and W. Lehner. *Data Mining in a Multidimensional Environment*. ADBIS 1999: 191-204
- [18] H. Cokrowijoyo, D. Taniar . *A framework for mining association rules in Data Warehouses* . IDEAL 2004: 159-165
- [19] S. Luján-Mora, J. Trujillo, I. Song: *Multidimensional Modeling with UML Package Diagrams*. ER 2002: 199-213

# Event-Feeded Dimension Solution

Tho Manh Nguyen<sup>1,2</sup>, Jaromir Neme<sup>1</sup>, and Martin Windisch<sup>1</sup>

<sup>1</sup> T-Mobile Austria, Rennweg 97-99, A-1030 Vienna, Austria  
{tho.nguyen, jaromir.nemec, martin.windisch}@t-mobile.at

<sup>2</sup> Institute of Software Technology and Interactive Systems, Vienna Uni. of Technology  
Favoritenstr. 9-11/188, A-1040 Vienna, Austria  
tho@ifs.tuwien.ac.at

**Abstract.** From the point of view of a data warehouse system its part of collecting and receiving information from other systems is crucial for all subsequent business intelligence applications. The incoming information can be classified generally in two types, the state-snapshot data and the state-change or event data usually called transactional data, which contains information about the change processes applied on the instances of information objects. On the way towards active data warehouses it becomes more important to provide complete data with minimal latency. We focus in this paper on dimensional data provided by any data-master application. The information transfer is done via messages containing the change-information of the dimension instances. The receiving data warehouse system is able to validate the event-messages, reconstruct the complete history of the dimension and provide a well applicable "comprehensive slowly changing dimension" (cSCD) interface for well-performing queries on the historical and current state of the dimension. A prototype implementation of "active integration" of a data warehouse is proposed.

## 1 Introduction

The upcoming integration technology standards [12,13] based on message exchange between information systems provide benefits not only to operative systems. Also non-OLTP systems like data warehouses can gain some profits out of this development [4]. In the past a restricted integration of the source systems traditionally led to batch-oriented data load approaches for data warehouses.

This situation is also in place at T-Mobile Austria, where we have done the analysis and development of the solution, proposed in this paper. The data warehouse at T-Mobile Austria runs on an Oracle 9.1.3 relational database and has a data volume of about six terabyte (TB). The complexity and number of operational source systems is very high. Therefore, the data warehouse provides its information as a single point of truth for nearly all units of the enterprise.

The data freshness for transactional data is very high, although the data is currently batch loaded. CDRs (call detail records) are usually loaded every four hours. The dimensional data loaded from legacies like the billing system (BSCS, Business Support & Control System), SAP or the CRM Systems is received via daily snapshots.

Because of the limitations of the snapshot based approach<sup>1</sup>, a more efficient approach being more near-real time is considered. Data changes in the operational sources are captured and provided near real time as event messages via the event-based infrastructure TIBCO [12]. This approach implies also the discussion of the message content and its validation. Mainly three quality aspects are under inspection: the completeness, uniqueness and order of the event-messages.

For the further processing of the event-messages we developed one comprehensive and general applicable SCD representation inspired by Kimball's three SCD-types [7] and propose a valid alternative to snapshot based information transfer. This solution is applicable especially in cases, where the information requirements of the receiving system focus on complete and detailed historical information for all instances combined with a minimal latency demand. For a given latency time-interval the advantages of this method are obvious for a dimension with a small number of changes compared to its cardinality.

The proposed method provides much more than a data replication. The primary target is not a physical mirror of the dimension object. Moreover all necessary views including the change history of this object are implemented in a standardized way with quite realistic efforts. The event feeded cSCD approach has been designed according to the main goals of T-Mobile Austria's data warehouse, which are simple: "to provide a single point of truth easy to access".

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces the Event Model concerning the event and event processing descriptions. The Event-feeded cSCD prototype implementation is described in Section 4. Finally, in Section 5, we present our conclusion and the future work.

## 2 Related Works

Active data warehouses [4,13] prefer to provide complete data within minimal latency. The well known limitations of processing dimensional snapshot-data [11] can be overcome by the proposed method, which is preferred for a dimension with a small number of changes compared to its cardinality because of less resource consumption. The complete history of the dimensional change events is also an advantage compared with historical (periodic) snapshots.

In some cases daily snapshots [1] have been used to provide change information out of the differences of two consecutive snapshots. To hold an appropriate history of such dimensions the only way was, to store the received snapshots chronologically, which means, that the storage request for the historical data does primarily not depend on the change fluctuation and volatility of the instances. One can apply in a second step some compression algorithms to overcome these disadvantages.

R. Kimball [7] has introduced the slowly changing dimension (SCD) types 1, 2 and 3 to track changes in dimension attributes. In SCD type 1, the changed attribute is

---

<sup>1</sup> Multiple change events between snapshots miss completely. For each snapshot comparison the number of records to process is high, requiring also high computing-resources and -time and the history is kept by tremendous daily snapshot versions consuming a lot of storage.

simply overwritten to reflect the most current value thus does not keep the historical changes. SCD type 2 creates another (dimension) record to keep trace the changed attributes, but could not keep the old value both before and after the change. For this purpose, the SCD type 3 uses the “current value” and “previous value” but it isn't appropriate for the unpredictable changes.

R. Bliujute et al. [2] suggested the temporal star schema to overcome the SCD issues with event and state-oriented data. They tackled the SCD type 2 in fixed attributes with timestamp. We instead propose a more general event model where the target dimensional object can be fine tailored depending on business requirements.

The flexibility in choosing the event timestamp (e.g. between transaction timestamp and processing timestamp) enables in the proposed cSCD representation the handling of time-consistency issues as discussed by R. Bruckner et al. [3].

J. Eder et al. [8] propose a temporal multi-dimensional data model to cope with the changes of dimension via multi versions and valid time. Our purpose is keeping track not only the versions of instances but also their relationships in dimensional data.

W. Inmon recommends the usage of normalized dimensions [6]. This is a very important aspect as the event based maintenance of denormalized dimensions although possible is not very practical [10].

### 3 Event Model

For a formal description of an event and event processing a UML based model is created. The core part of this model is an UML profile describing the event meta-model. Additionally, the semantic of event is defined and shown by a simple example. Possible strategies of event interpretation are discussed.

Based on the defined model and the interpretation of the event a broader discussion is performed, demonstrating that the traditional distinction between fact and dimension in event based DWH environment represents only a different specialization of our event based model.

#### 3.1 Profile

To describe a general event it is necessary to raise the model to the meta level M2 [9] as the structure of each event type is very proprietary based on the transferred business information. The simplified profile definition is depicted in Fig. 1.

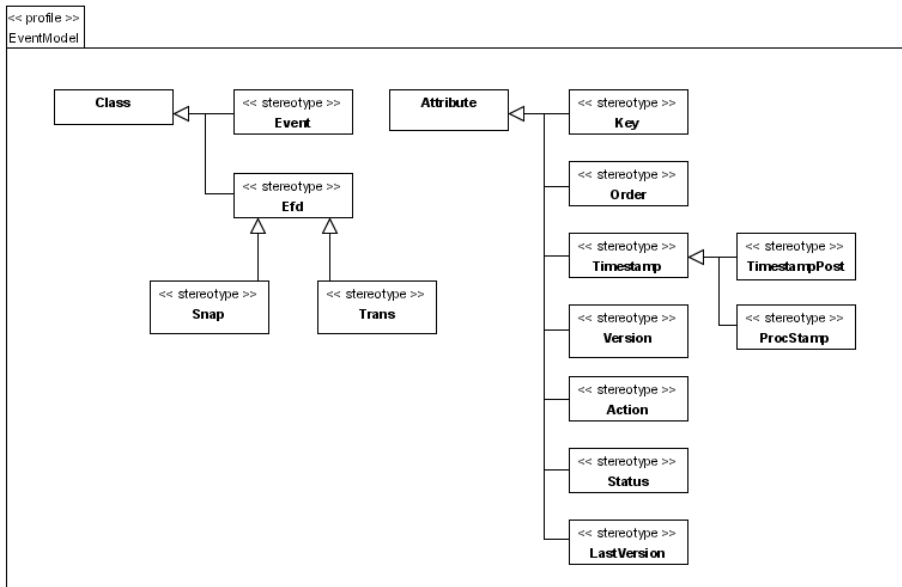
The key concepts of the event profile are as follows:

- Stereotype «Event» describes the object containing the event data.
- Stereotype «Efd» (abbreviation for event feeded dimension) depicts the target object that is maintained via the event stream.
- Stereotypes «Trans» and «Snap» as subtypes of «Efd» are discussed below

Those stereotypes are applicable on the class level, the rest of stereotypes are connected with an attribute:

- Stereotype «Key» is used to mark the (natural or surrogate) primary key of the dimension

- Stereotype «Order» is intended to define the order in which the events were created and should be processed.
- Stereotype «Timestamp» identifies an attribute containing the timestamp information of an event. The transaction time, event creation time, event processing time are various examples of this stereotype.
- Stereotype «Action» describes the nature of the change represented in the event (insert / update / delete)
- Stereotype «Status» enables the depiction of a logical deletion of a dimension instance.



**Fig. 1.** Simplified Profile Event Definition

Not all of the listed stereotypes are mandatory, the usage is constrained by semantic rules (see below) such as: The «Event» and «Efd» classes must contain at least one *Key* attribute (i.e. an attribute with stereotype «Key»). *Order* and *Timestamp* attributes may coincident, e.g. in cases when the time grain is too large to distinct the events uniquely, the *Order* attribute is used to define the unique event sequence. The opposite extreme when neither of those attributes is defined is also valid. In that case the timestamp of event processing can be used as a default *Timestamp* attribute (of course the unique order of events must be established in this case as well).

### 3.2 Example

To illustrate the usage of Event profile let's consider a simplified application that maintains the customer attributes via an event interface. The customer is identified with an attribute id, the customer attributes consist of name, address and tariff.

The class with associated stereotype *Event* describes the customer-value-change event. This event is generated on each change of at least one attribute of a particular customer. As marked with stereotype *Key* the primary key of the customer dimension is the attribute *id*. The attribute *timestamp* is stereotyped as *Timestamp* i.e. this attribute defines the point in the time of the change of customer attributes. The rest of attributes have no stereotypes they are regular event attributes containing additional information.

The second class in Fig. 2 describes the target object maintained via the event feed (stereotype *Trans* defines that a full versioned history of the target object will be build; see the detailed discussion in 3.3). The meaning of the additional attribute is discussed below.

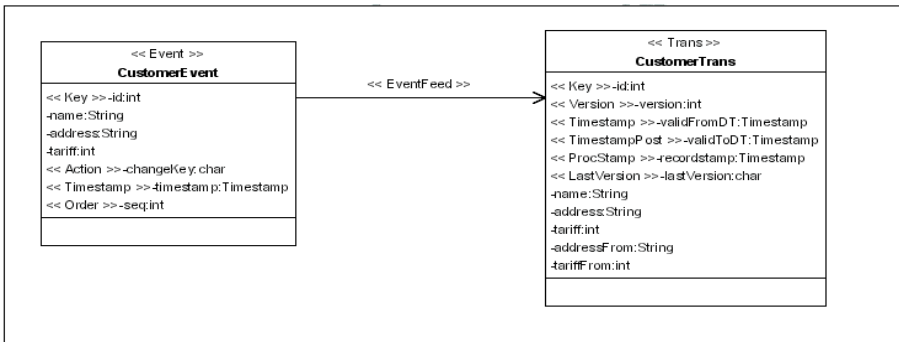


Fig. 2. Customer Event Profile Example

### 3.3 Event Processing

The profile based event model must be enriched with semantic rules defining the interpretation of an event. The most important feature is the sub-typing of the *Efd* object. In the profile two main examples are defined *Snap* and *Trans*.

The *Snap* object is maintained with overwrite policy, i.e. new records are inserted; existing records are updated or deleted. In a *Snap* object only one record per primary key is stored. *Snap* is mnemonic for dimension snapshot.

The *Trans* object is maintained cumulatively, each event is added to the target object, building a complete transactional history of the dimension.

The handling of primary key of the build dimension can be configured. The Primary key option defines if the target object uses the natural key (as provided within the event) or if a surrogate key should be generated while the event is processed. In any case the primary key always uniquely identifies the dimension instance, so if a complete history of the dimension is maintained an additional attribute stereotyped as «Version» must extend the primary key of the target table.

Other option is defined on the level of attribute; an attribute noted as *Timestamp-Post* is applicable for *Trans* object only. It is filled with the value of the corresponding *Timestamp* attribute of the successor version decreased by the smallest grain of



the time dimension (e.g. 1 ms). The default value is an artificially set high date (e.g. 31-12-9999 00:00:00). The usage of two timestamps in a full history table is not a "pure relational" design but extreme practical solution as for the selection of a version of a particular dimension occurrence a simple logic can be applied (*required timestamp* between *Timestamp* and *TimestampPost*).

If an attribute has a suffix *From* it contains the value "before the change", i.e. in *Trans* object this is the value stored in the preceding version. The association between the corresponding attribute is established with naming conventions.

A different aspect of semantic is the validation of the event model, i.e. the verification if the model is well-formed. Examples of constraints that must be checked are listed below:

- Event class must have at least one *Key* attribute
- Each *Timestamp* attribute must have a type compatible with date/time.

The final role of semantic in event context is the event validation. It is possible to extend the event data with redundant information that can be checked while the event is processed. The exceptions can be interpreted as an advice of lost or corrupted events.

For examples adding an *Action* attribute to the event (possible values: insert / update / delete) enables additional checks:

- key must exists in the target object on update and delete
- key must not exists in the target object on insert

Other types of validation can be alternatively implemented as services on the event transport layer, e.g. guaranteed delivery or de-dup filtering [5].

## 4 Event-Feeded cSCD Implementation

The described implementation represents a particular instantiation of the presented model in Section 3. The target object is implemented as a *Trans* table; natural key option is used; *Action* and *from attributes* are supported.

### 4.1 Development Environment

Because the target DWH is also based on Oracle DBMS, we decided to keep the current development environment, i.e. developing the event feed cSCD solution as an Oracle PL/SQL package and call easily the functionality from ETL (e.g. Informatica Powercenter) mappings.

### 4.2 The UTL\_EVENT\_SCD Package

The package can be used to trace the changing attributes of any (dimensional) table. It accepts a variant of parameters for the detailed configuration of the event-processing and -correction such as *traced entity* (via table name parameter), *correct option* (optional, mandatory or automatically), *refresh option* (incremental or from scratch), *filer criteria*, etc.

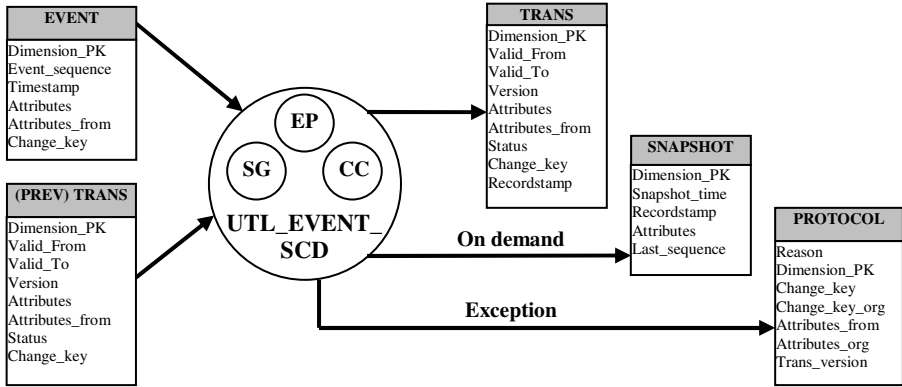


Fig. 3. UTL\_EVENT\_SCD Package modules and its related tables

The package (Fig. 3) contains 3 main modules: Event Processing (EP), Snapshot Generation (SG), and Consistency Checking (CC) providing the following options:

- Validating the events before refreshing the historical transactions of the entity instances (update TRANS table) with full historical tracing and versioning.
- Providing the state information at any point in time for any instance or subset of instances (generate SNAPSHOT table on demand)
- Checking the consistency between the entity state data of the legacy system and the data in DWH, and solving the inconsistency issue.

#### 4.2.1 Event Processing (EP)

EP applies event data and refreshes the TRANS table as follows. It first accesses the event data, filters those which happened since the last refresh (i.e. those records which do not appear in TRANS or have different states with the current records in TRANS). The event validation then checks the events with automatic correction options to override some invalid events. This validation and correction processes are based on some useful attributes such as *change\_key*, *attribute\_from* or *sequence order*. The invalid or overridden events are kept in the PROTOCOL table.

The valid events are used to refresh the TRANS table. For each event data related to an entity instance, an equivalent transaction record in TRANS table is created. If there are other events related to the same entity, the extended SCD type 2 [1,5] is applied to keep trace over all transactions (with versions). The TRANS table thus contains the complete transaction history of dimension changes.

Examples: We apply the UTL\_EVENT\_SCD package to trace the Customer’s attribute changes. Suppose that we have currently two customers Robert and Sonja until 7 am,14/02/2005. At 7:10, Robert informs that he changes his address from 20 Rennweg to 25 Favoriten. 7:12 am, a new customer Micheal has registered into the system, and Robert changes his tariff from type 1 to type 2 at 7:13. At 7:14, Sonja changes her tariff from type 2 to type 1. The UTL\_EVENT\_SCD package is executed at 7:15 to refresh the previous TRANS table. (Fig. 4)

CUST_TRANS (Before Event applying)												
ID	Valid from	Valid to	Name	Address	Tariff	Address_from	Tariff_from	Recordstamp	version	Last_version	Change_key	
1	14-02-2005 07:00:00	31-12-9999 00:00:00	Robert	20 Rennweg	T1			14-02-2005 07:00:00	1	Y	I	
2	14-02-2005 07:00:00	31-12-9999 00:00:00	Sonja	15 Kargan	T2			14-02-2005 07:00:00	1	Y	I	

CUST_EVENT								
ID	Timestamp	Seq	Name	Address	Tariff	Address_from	Tariff_from	Change_key
1	14-02-2005 07:10:00	1	Robert	25 Favoriten	T1	20 Rennweg		U
3	14-02-2005 07:12:00	2	Micheal	10 Rathaus	T2			I
1	14-02-2005 07:13:00	3	Robert	25 Favoriten	T2		T1	U
2	14-02-2005 07:14:00	4	Sonja	15 Kargan	T1		T2	U

CUST_TRANS (After Event applying)												
ID	Valid from	Valid to	Name	Address	Tariff	Address_from	Tariff_from	Recordstamp	version	Last_version	Change_key	
1	14-02-2005 07:00:00	14-02-2005 07:09:59	Robert	20 Rennweg	T1			14-02-2005 07:15:00	1	N	I	
1	14-02-2005 07:10:00	14-02-2005 07:12:59	Robert	25 Favoriten	T1	20 Rennweg		14-02-2005 07:15:00	2	N	U	
1	14-02-2005 07:13:00	31-12-9999 00:00:00	Robert	25 Favoriten	T2		T1	14-02-2005 07:15:00	3	Y	U	
2	14-02-2005 07:00:00	14-02-2005 07:13:59	Sonja	15 Kargan	T2			14-02-2005 07:15:00	1	N	I	
2	14-02-2005 07:14:00	31-12-9999 00:00:00	Sonja	15 Kargan	T1		T2	14-02-2005 07:15:00	2	Y	U	
3	14-02-2005 07:12:00	31-12-9999 00:00:00	Micheal	10 Rathaus	T2			14-02-2005 07:15:00	1	Y	I	

Fig. 4. TRANS table refresh after UTL\_EVENT\_SCD package execution

The investigation of the performance behavior based on the prototype implementation showed a near linear scalability of the processing-time per event with an average throughput of about 300 TRANS-records per second on a dimension with the cardinality of one million records. The minimum refresh period is about 3-4 seconds caused by process overheads. However, with the high number of events (e.g. over 20000 events), the more events accumulated, the less efficient of the event-SCD approach compared to the snapshot based SCD approach.

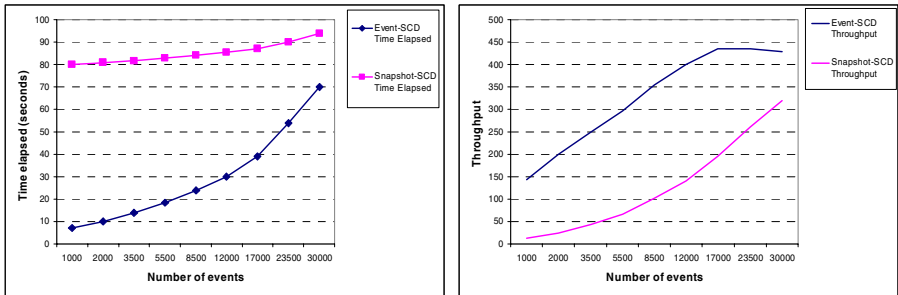


Fig. 5. Elapsed processing time and performance throughput comparison between event-SCD and snapshot based SCD approach

### 4.2.2 On Demand Snapshot Generation (SG)

Despite the series of snapshots is not kept as previously, the requirement to have a snapshot at one point in time for any subset of entity instances remains. From the TRANS table, we can rebuild these required snapshots. The package provides two options to generate a snapshot: (1) from scratch (Fig. 6) and (2) based on an existing snapshot, further referenced as based snapshot (Fig.7). The generated Customer snapshots at 7:00 and 7:15 are shown in Fig. 8.

```
CREATE TABLE CUST_SNAP AS
SELECT ID, i_timepoint as Snaptime, Name,Address, Tariff
FROM CUST_TRANS
WHERE CHANGE_KEY <> 'D' AND
i_timepoint BETWEEN VALIDFROM_T AND VALIDTO_T;
```

**Fig. 6.** Create Snapshot from scratch (i\_timepoint is the time point of the snapshot data)

```
CREATE TABLE CUST_SNAP AS
SELECT * FROM
(SELECT ID,i_timepoint as Snaptime, Name, Address, Tariff
FROM CUST_TRANS WHERE CHANGE_KEY <> 'D' AND
i_timepoint BETWEEN VALIDFROM_T AND VALIDTO_T
AND VALIDFROM_T > v_prev_time
UNION ALL
SELECT ID,i_timepoint as Snaptime, Name, Address, Tariff
FROM BASED_CUST_SNAP
WHERE ID NOT IN
(SELECT ID FROM CUST_TRANS WHERE
i_timepoint BETWEEN VALIDFROM_T AND VALIDTO_T
AND VALIDFROM_T > v_prev_time)
);
```

**Fig. 7.** Create Snapshot from based snapshot (BASED\_CUST\_SNAP is the based snapshot table, v\_prev\_time is the time point of the based snapshot data)

SNAPSHOT at 14-02-2005 7:00			
ID	Snaptime	Name	Tariff
1	14-02-2005 07:00:00	Robert	T1
2	14-02-2005 07:00:00	Sonja	T2

SNAPSHOT at 14-02-2005 7:15			
ID	Snaptime	Name	Tariff
1	14-02-2005 07:15:00	Robert	T2
2	14-02-2005 07:15:00	Sonja	T1
3	14-02-2005 07:15:00	Micheal	T2

**Fig. 8.** SNAPSHOT tables generated at 7:00 and 7:15

### 4.2.3 Consistency Checking and Recovery (CC)

In the event based cSCD approach, an inconsistent state could be detected when we are able to access on a truthful snapshot source (usually provided from the legacy systems). The input requirements of this process are the mandatory truthful snapshot ( $S_i, t_j$ ) table and the metadata parameters describing the record-structure. The consistency checking process compares a truthful snapshot(-part) taken on any subset of instances  $S_i$ , at any point of time  $t_j$  with the corresponding on demand snapshot ( $S_i, t_j$ ) (see Section 4.2.2) which is temporary stored in a TEMP\_SNAP ( $S_i, t_j$ ) table. The found inconsistencies between the snapshots are applied again as new change events to correct the TRANS records.

## 5 Conclusion and Future Work

The event feeded cSCD approach presented in this paper significantly reduces the number of records to be processed compared to the snapshot based approach. Besides, compared with the Kimball's classification of SCD [7] we see that the SDC types 1,2

and 3 are only examples of possible instantiations of the proposed cSCD approach (SDC 1 and 2 respectively use the *Snap* object without and with *from attributes*; SDC 3 is based on *Trans* object without *from attributes*).

Although the target object was up to now considered as a dimension, this is not a limitation of the proposed model. A typical fact table can be described also as a versioned dimension (fast changing dimension), using the add-version update policy (each event creates a new record in the fact table) with appropriate validation e.g. to maintain a balance attribute.

Further more extending our model with summarizing stereotypes (e.g. add the actual value of the attribute to the previous value) the way is opened for describing running aggregates. On the other hand the correlation of system-dependent event-messages as an alternative to the join of dimensional snapshots needs further inspection.

## Acknowledgement

This research was funded by T-Mobile Austria and supported by the IT department providing the needed infrastructure and environment.

## References

1. Arun Sen, Atish P. Sinha, A Comparison of Data Warehousing Methodologies. Communications of the ACM, Vol. 48, No. 3, March 2005.
2. Bliujute, R., Saltenis, S., Slivinskas, G., and Jensen, C.S. (1998). Systematic Change Management in Dimensional Data Warehousing. In Proc. of the 3rd Intl. Baltic Workshop on Databases and Information Systems , Riga, Latvia, (pp. 27-41).
3. Bruckner R., Tjoa A., Managing Time Consistency for Active Data Warehouse Environments. DaWaK 2001, Springer-Verlag LNCS 2114, pp. 254–263, 2001.
4. Brobst, S., Enterprise Application Integration and Active Data Warehousing, In Proc. Data Warehousing 2002, pp. 15-22, Physica-Verlag 2002.
5. Hohpe G., Woolf B., Enterprise Integration Patterns, Designing, Building, and Deploying Messaging Solutions, Addison-Wesley, 2004
6. Inmon, W., Building the Data Warehouse; Jon Wiley & Sons, Second Edition, 1996
7. Kimball R. et al., The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd Edition, John Wiley & Sons, 2002.
8. Koncilia, C., Eder, J., Changes of Dimension Data in Temporal Data Warehouses, DaWaK 2001, Springer-Verlag LNCS 2114, pp. 284–293, 2001.
9. Meta Object Facility (MOF) Specification <http://www.omg.org/docs/formal/00-04-03.pdf>
10. Rieger B., Brodmann K., Mastering Time Variances of Dimension Tables in the Data Warehouse, Osnabrueck University, 1999
11. Rocha R., Cardoso F., Souza, M., Performance Tests in Data Warehousing ETL Process for Detection of Changes in Data Origin. DaWaK 2003, LNCS 2737, pp. 129-139, 2003.
12. TIBCO Software Inc.: <http://www.tibco.com>
13. Vandermay J., Considerations for Building a Real-time Oracle Data Warehouse, DataMirror Corporation White Paper, 2000.

# XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses

Byung-Kwon Park<sup>1</sup>, Hyoil Han<sup>2</sup>, and Il-Yeol Song<sup>2</sup>

<sup>1</sup> Dong-A University, Busan, Korea  
bpark@dau.ac.kr

<sup>2</sup> Drexel University, Philadelphia, PA 19104, USA  
hyoil.han@cis.drexel.edu, songiy@drexel.edu

**Abstract.** Recently, a large number of XML documents are available on the Internet. This trend motivated many researchers to analyze them multi-dimensionally in the same way as relational data. In this paper, we propose a new framework for multidimensional analysis of XML documents, which we call *XML-OLAP*. We base XML-OLAP on XML warehouses where every fact data as well as dimension data are stored as XML documents. We build XML cubes from XML warehouses. We propose a new multidimensional expression language for XML cubes, which we call *XML-MDX*. XML-MDX statements target XML cubes and use XQuery expressions to designate the measure data. They specify text mining operators for aggregating text constituting the measure data. We evaluate XML-OLAP by applying it to a U.S. patent XML warehouse. We use XML-MDX queries, which demonstrate that XML-OLAP is effective for multi-dimensionally analyzing the U.S. patents.

## 1 Introduction

An online analytical processing (OLAP) system is a powerful data analysis tool for decision-making [11]. It provides an analysis from multiple perspectives or dimensions for a large amount of data residing in a data warehouse. Data warehouses are commonly organized with one large fact table and multiple small dimension tables. The fact and dimension tables are typically the structured data stored in a relational database.

Recently, a large number of XML documents are available on the Internet. Thus, we need to analyze them multi-dimensionally in the same way as relational data. However, the data model of XML documents is not flat like relational data, but a tree structure. In addition, XML documents can contain unstructured data such as text. Thus, we need to develop a new framework of multidimensional analysis for XML documents.

In this paper, we propose an OLAP framework for XML documents, which we call *XML-OLAP*. We base XML-OLAP on XML warehouses where every fact data as well as dimension data is stored as an XML document. XML cubes are built from XML warehouses. While conventional data cubes have numeric data as measure values, XML cubes have either numeric or text data. We propose a

new multidimensional expression language over XML cubes, which we call *XML-MDX*. XML-MDX statements target XML cubes and use XQuery expressions to specify the measure data, axis dimensions, and slicer. They also specify text mining operators for aggregating text constituting the measure data such as summarization, classification, and top keyword extraction.

We show an XML-OLAP example for the U.S. patent warehouse to evaluate its effectiveness. The U.S. patent warehouse is built by extracting information from the U.S. Patent Web Site [12], converting it into XML documents, and storing them in a native XML database. Dimension tables are also built in the form of XML documents and stored in the native XML database. We demonstrate XML-MDX queries to show that XML-OLAP is effective for multi-dimensionally analyzing the U.S. patents.

This paper makes the following contributions: (1) We propose a new framework, XML-OLAP, for multi-dimensional analysis of XML documents. We believe XML-OLAP is the first framework for online analysis of an XML document set. (2) We propose a new multidimensional expression language, XML-MDX. We are inspired by the Microsoft MDX language [11] which is widely accepted as an OLAP query language. XML-MDX can accommodate the hierarchical tree structures of XML documents. (3) We propose a mechanism to enable the aggregation of text data contained in XML documents using text mining operations. It can give the text mining community a vehicle to make their technology accessible to a broad user base.

This paper is organized as follows: Section 2 describes the related work. Section 3 describes building an XML warehouse. Section 4 describes building XML cubes and querying them using XML-MDX. Section 5 describes the XML-OLAP application to the U.S. patent XML data. Section 6 concludes the paper.

## 2 Related Work

Pokorny [9] applied a star schema to XML data. A dimension hierarchy is defined as a set of logically connected collections of XML data. Facts may also be conceived as elements of XML data. Pokorny proposed a formal model for dimension hierarchies and referential integrity constraints in an XML environment. We also assume that both dimension and fact information are represented as XML documents in the same way as Pokorny does.

Nassis et al. [7] also worked on XML document warehousing. They focused on the conceptual design of XML document warehouses and the concept of virtual dimensions using XML views. They utilized object-oriented concepts in UML to develop a conceptual model for XML document warehouses from user requirements.

Golfarelli et al. [2] dealt with the problem of automatically deriving the conceptual schema from an XML source. They assumed that the XML data have all the information for the schema. They proposed a semi-automatic approach for building a schema from an XML DTD or schema.

Hümmer et al. [3] proposed a family of XML document templates, called XCube, to describe a multidimensional structure, dimensions and fact data for integrating several data warehouses into a virtual or federated data warehouse. The XML templates are not directly related to XML warehousing, but they can be used for representing hierarchical dimension data in our framework.

There are a lot of work on constructing OLAP cubes from distributed XML data. Jensen et al. [4,5] transformed XML data on the web into relational data in order to be used by conventional OLAP tools. Niemi et al. [8] proposed a system which can construct an OLAP cube based on an user's MDX query. They all construct a relational OLAP cube by transforming XML data collected from distributed XML sources, whereas we construct an XML cube from XML documents.

### 3 XML Warehouses

In Section 3.1, we present the multidimensional model of an XML warehouse and how to derive it. In Section 3.2, we present how to build an XML warehouse from the given XML document set.

#### 3.1 Multidimensional Modeling of XML Warehouses

We assume that an XML warehouse has a multidimensional model as in Figure 1. The model has a single repository of XML documents, which forms fact data, and multiple repositories of XML documents, in which each forms one dimension data. In Figure 1, there are  $n$  dimensions and thus,  $n$  repositories of XML documents.

The fact repository is the same as assumed by Nassiss et al. [7]. Each fact is described in a single XML document. Thus, the fact data is not as simple as in a conventional data warehouse. It has a hierarchical tree structure containing structured data and unstructured data.

Dimension data are described in XML documents, and each dimension data is grouped into a repository of XML documents. Since each dimension has a hierarchy, a single XML document in a repository contains an instance of a dimension hierarchy rooted at a top level member in the hierarchy. Some auxiliary data structures like indexes are used to link dimension data with fact data.

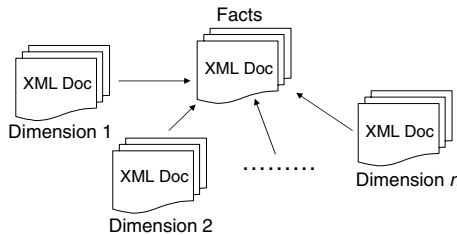


Fig. 1. Multidimensional Model of An XML Warehouse



The XML multidimensional model described in this section has the following advantages: (1) Since all the fact and dimension data are described in XML documents, we can easily collect them. (2) We can store all the fact and dimension data in a native XML database and can easily manage and query them through the native XML database management system. (3) Since each dimension data is described in an XML document, we can easily represent the dimension hierarchy in a single document using the tree structure of an XML document. Thus, we need not join multiple tables as required in the relational snowflake model.

### 3.2 Building an XML Warehouse

Building an XML warehouse consists of two steps: building a single XML repository for fact data and building a number of XML repositories for dimension data. Rusu et al. [10] dealt with the problem of processing raw XML documents into a data warehouse repository, and proposed some rules and techniques for data cleaning, integration, summarization, and updating/linking existing documents. We assume that the XML repository for fact data is provided after cleaning. We focus on building the XML repositories for dimension data. In order to decide the required dimensions to analyze the given XML document repository, we need to build the conceptual model of the XML documents.

There are several works on the conceptual modeling of XML data using UML. Jensen et al. [5] proposed an algorithm for automatically constructing UML diagrams from XML data based on their DTD's. Lujan-Mora et al. [6] extended UML for multidimensional modeling including multistar model, shared hierarchy levels, and heterogeneous dimensions. We adopt their methods for conceptual modeling of XML data in UML class diagrams.

From the conceptual model of fact data, we can decide dimensions for analyzing them. We assume that it is done manually because the conceptual model is object-oriented and expressed in UML class diagrams for the purpose of helping people to understand the logical structure of fact data. Nassis et al. [7] proposed to select dimensions based on user requirements and to represent the dimensions virtually using XML views since they assumed that all the dimension data are part of fact data. However, we assume that some dimension data are out of fact data. Thus, for simplicity, we materialize an XML repository for each dimension selected.

For multidimensional analysis, we need some mechanism to join dimension and fact data. In a conventional data warehouse, we insert a foreign key in the fact data to match with each dimension data. In this paper, we rely on an index structure that matches each dimension data with the corresponding fact data. We build the index together with the XML repositories for dimension data.

## 4 The Multidimensional Analysis Framework for XML Warehouses

In this section, we describe XML-OLAP, which is about generating XML cubes and expressing multidimensional queries. In Section 4.1, we present a new notion

of XML cube (called *XQ-Cube*). In Section 4.2, we present a new multidimensional expression language (called *XML-MDX*) for XQ-Cubes.

#### 4.1 XML Cubes

Since our XML warehouse has XML documents as fact data, the cube constructed from the XML warehouse should have the cells whose values are an aggregation of XML documents. Defining an aggregation over multiple XML documents is difficult because an XML document is a hierarchically structured composite data object. However, defining an aggregation over such data segments as numeric or text data segments is possible.

We propose to use an XQuery expression for measure specification. We call the cube constructed from the XML warehouse with measure values described by an XQuery expression *XQ-Cube*. The measure values, the evaluation result of the XQuery expression, are numeric or text data. If they are numeric, the aggregation will be the same as in relational cubes such as addition and average; otherwise, the aggregation will be a kind of text operation. We introduce text mining operations for text aggregation (see Section 4.2).

An XQ-Cube has the following advantages: (1) A variety of cubes can be generated. Since the measure data is specified using an XQuery expression, any kind of cube can be defined that XQuery can generate. (2) It provides an aggregation mechanism over XML documents. Since the measure data is a fragment of an XML document, we can apply various aggregation operators according to the data type. (3) An XQ-Cube is a generalization of a relational cube. It becomes a relational cube when the XQuery expression is evaluated to numeric data, while it becomes a text cube when evaluated to text data.

#### 4.2 Multidimensional Expression Language

For querying a cube, we need a query language for cubes. Microsoft designed a multidimensional expression language called *MDX* [11] for relational cubes. We are inspired by Microsoft MDX to design a new multidimensional expression language called *XML-MDX* for XQ-Cubes. XML-MDX has two statements: *CREATE XQ-CUBE* and *SELECT*. The former is for creating a new XQ-Cube, and the latter for querying. In general, in order to enhance the query processing performance of XML-MDX or to use an XQ-Cube multiple times, we first create an XQ-Cubes and then refer to it in queries.

*CREATE XQ-CUBE*: Figure 2 shows the basic syntax of the *CREATE XQ-CUBE* statement. The  $\langle \text{XQ-Cube\_name} \rangle$  value specifies the name of the XQ-Cube to create. A *CREATE XQ-CUBE* statement is composed of two clauses: *FROM* and *WHERE*. The created XQ-Cube is stored for use by XML-MDX queries.

The *FROM* clause specifies the measure data from which the XQ-Cube will be created. Figure 3 shows the definition of the *FROM* clause expressed in BNF notation. The  $\langle \text{XQ-Cube\_specification} \rangle$  value specifies the measure data using

```
CREATE XQ-CUBE <XQ-cube name>
FROM <XQ-cube specification>
[ WHERE < slicer specification > ]
```

**Fig. 2.** Definition of CREATE XQ-CUBE statement in BNF

```
<FROM_clause> ::= FROM <XQ-cube_specification>
<XQ-cube_specification> ::= <XQuery_expression> : <aggregation_operator> ]
<aggregation_operator> ::= ADD | LIST | COUNT | SUMMARY | TOPIC |
TOP KEYWORDS | CLUSTER
```

**Fig. 3.** Definition of FROM Clause in BNF

```
<WHERE_clause> ::= WHERE < slicer_specification >
< slicer_specification > ::= (“ <XQuery_expression> { “,” <XQuery_expression> } “”)
```

**Fig. 4.** Definition of WHERE Clause in BNF

```
SELECT <axis 0 specification>,
      <axis 1 specification>,
      ...
FROM <XQ-Cube name>
[ WHERE < slicer specification > ]
```

**Fig. 5.** Basic Syntax of XML-MDX

an XQuery expression. We should specify an aggregation operator according to the measure data, the evaluation result of the XQuery expression.

In this paper, we define the following seven aggregation operators: ADD, LIST, COUNT, SUMMARY, TOPIC, TOP KEYWORDS, and CLUSTER. ADD is for numeric data as it is in relational OLAP. The other operators are all for non-additive data including text. LIST displays the data consecutively in a sequence. COUNT displays the number of measure data. The others are all from text mining techniques: SUMMARY, TOPIC, and TOP KEYWORDS display a total summary, a topic, and top keywords respectively from all the text data to aggregate. CLUSTER builds a cluster over all the text data to aggregate. We can expand the aggregation operators as the text mining techniques progress.

The WHERE clause is optional. It determines which dimension members to use for the slicer which restricts the extractions of data to the determined dimension members. Figure 4 shows the definition of the WHERE clause expressed in BNF notation. The < slicer\_specification > value specifies a slicer which is a tuple of XQuery expressions. Each XQuery expression specifies a dimension member which filters off the other members. A special XQuery expression is used for specifying the 'All' member (see Figure 8). The dimensions that are not specified in the < slicer\_specification > form the axis dimensions of the XQ-Cube created.

```

<SELECT_clause> ::= SELECT <axis_specification> { "," <axis_specification> }
<axis_specification> ::= <XQuery_expression_set> ON <axis_name>
<XQuery_expression_set> ::= "{" <XQuery_expression> { "," <XQuery_expression> } "}"
<axis_name> ::= COLUMNS | ROWS | PAGES | SECTIONS | CHAPTERS |
                AXIS(<index>)

```

**Fig. 6.** Definition of SELECT Clause in BNF

*SELECT*: Figure 5 shows the basic syntax of the SELECT statement. A SELECT statement has the same structure as that of Microsoft MDX, which is composed of three clauses: SELECT, FROM, and WHERE. The FROM clause designates an XQ-Cube name previously created by a CREATE XQ-CUBE statement. We populate the result set of the SELECT statement from the designated XQ-Cube.

The SELECT clause specifies axis dimensions. Each axis dimension determines an edge of a multidimensional result set. Figure 6 shows the definition of the SELECT clause expressed in BNF notation. Each <axis\_specification> value defines one axis dimension. The number of dimensions in an XML warehouse is the maximum number of axis dimensions. An <axis\_specification> value is broken down into a set of XQuery expressions and an axis name.

The result set of the XQuery expressions constitute the members of an axis dimension. Since each dimension having a hierarchical structure is represented in a single XML document, an XQuery expression specifies a member of a dimension level. We need an XQuery expression for each member of an axis dimension.

We assign axis names in the same way as Microsoft MDX does [11]. Each axis dimension is associated with a number: 0 for the X-axis, 1 for the Y-axis, 2 for the Z-axis, and so on. The <index> value is the axis number. For the first 5 axes, the aliases COLUMNS, ROWS, PAGES, SECTIONS, and CHAPTERS can be used in place of AXIS(0), AXIS(1), AXIS(2), AXIS(3), and AXIS(4), respectively. An XML-MDX query cannot skip axes. That is, a query that includes one or more axes must not exclude lower-numbered or intermediate axes.

The definition of the WHERE clause of the SELECT statement is the same as that of CREATE XQ-CUBE statement. The < slicer\_specification > value filters the XQ-Cube specified in the FROM clause. Note that, as in Microsoft MDX, the dimensions that are not explicitly assigned to axes in the SELECT clause are assumed to be slicer dimensions. They filter the XQ-Cube with their default members. A default member is the All member if an 'All' level exists, or an arbitrary member of the highest level.

XML-MDX has the following advantages over Microsoft MDX: (1) It can have all the features of Microsoft MDX since XML-MDX is designed based on Microsoft MDX. (2) Composing and processing XML-MDX queries are easy since all the specifications for the measure values and the dimension members are expressed in XQuery. We use an existing XQuery engine to process XML-MDX queries since no special syntax is required for XML-MDX. (3) When specifying slicer or axes, selecting the dimension members satisfying a condition is possible since we are using XQuery to specify them. Microsoft MDX has no such facility and use only a path in the dimension hierarchy.

## 5 Application to US Patent XML Warehouse

We assume that we are given a huge collection of XML documents about U.S. patents. They form the XML repository representing fact data of the U.S. patent XML warehouse. Figure 7 shows an example of such documents. After reviewing the fact repository, we build a conceptual model using a UML class diagram. From the conceptual model, we decide the dimensions to use for multidimensional analysis.

Figure 8 shows the hierarchies of the four dimensions selected for the U.S. patent XML warehouse. Each dimension has the 'All' level. The two dimensions, 'Appl.Time' and 'Reg.Time', represent when a patent was applied and registered respectively. The dimension, 'Inventor', represents a patent's inventors. The dimension, 'Topic', represents a patent's classification.

Figure 9 shows an XML document in the XML repository representing the dimension, 'Appl.Time'. The document is about an application year, 1998. The

```

<uspatent>
  <title>
    <text> Rule based database security system and method </text>
  </title>
  <abstract>
    <text> A rule-based database security system and method are disclosed. </text>
  </abstract>
  <inventor>
    <name> Cook; William R. </name>
    <addr> Redwood City, CA </addr>
  </inventor>
  <patent>
    <no> 6,820,082 </no>
    <applNo> 541227 </applNo>
  </patent>
  <registeredOn> <date> November 16, 2004 </date> </RegisteredOn>
  <filedOn> <date> April 3, 2000 </date> </FiledOn>
  <claim>
    <number> 1 </number>
    <text> A method for processing requests from a user to perform an act ...</text>
  </claim>
</uspatent>

```

Fig. 7. Fact Data about U.S. Patents

Appl. Time	Reg. Time	Inventor	Topic
All	All	All	All
Year	Year	Inst.Type	High
		Institute	Middle
Month	Month		Low
		Inventor	

Fig. 8. Dimension Hierarchies of U.S. Patent XML Warehouse

```

<year num = "1998">
  <month num = "3" name = "Mar." />
  <month num = "9" name = "Sep." />
</year>

```

Fig. 9. An XML Document for Dimension *ApplTime*

```

<instType name = "university" code = "001">
  <institute name = "Drexel" addr = "Philadelphia, PA">
    <inventor name = Il-Yeol Song" addr = "Philadelphia, PA" />
  </institute>
</instType>

```

**Fig. 10.** An XML Document for Dimension *Inventor*

```

CREATE XQ-CUBE XQ-Cube-1
FROM col('/db/uspatent')/patent/no : COUNT
WHERE ( col('/db/applTime')/ALL,
        col('/db/regTime')/year[@num>2000] )

```

**Fig. 11.** An Example of CREATE XQ-Cube Statement

level, 'year', has an attribute, 'num', It has the lower level, 'month', having two attributes: 'num' and 'name'. The level, 'month', has two members whose values of the attribute, 'num' are 3 and 9 respectively.

Figure 10 shows an XML document in the XML repository representing the dimension, 'Inventor'. The document is about an institution type, 'university', which is a member of the level, 'instType' of the dimension. The level, 'instType', has two attributes: 'name' and 'code'. It has the lower level, 'institute', having two attributes: 'name' and 'addr'. The level, 'institute', has the lower level, 'inventor', having two attributes: 'name' and 'addr'.

Figure 11 shows an example to create an XQ-Cube named XQ-Cube-1. The XQuery expression, "col('/db/uspatent')/patent/no", specifies the measure of XQ-Cube-1. The collection, "/db/uspatent" contains the fact data from which we collect "/patent/no" for the measure. The aggregation operator, COUNT, counts the number of patent no's. The WHERE clause has two XQuery expressions. The collection, "/db/applTime" contains the XML documents for 'Appl.Time'. The special XQuery expression, "/All" means that the 'All' level is selected for slicing, which results in the aggregation along all the members of the dimension. The collection, "/db/regTime" contains the XML documents for 'Reg.Time'. The XQuery expression, "/year[@num>2000]" results in slicing off all 'year' less than or equal to 2000. As a result, XQ-Cube-1 has three axis dimensions: 'Inventor', 'Topic', and 'Reg.Time' with 'year' greater than 2000.

Figure 12 shows an XML-MDX query for XQ-Cube-1. The slicer specification in the WHERE clause slices off the registration years less than or equal to 2002.

```

SELECT { col('/db/topic')/high[@topic='XML'],
        col('/db/topic')/high[@topic='OLAP'] } ON COLUMNS
      { col('/db/inventor')/instType[@name='university'],
        col('/db/inventor')/instType[@name='industry'] } ON ROWS
FROM XQ-Cube-1
WHERE ( col('/db/regTime')/year[@num > 2002] )

```

**Fig. 12.** An Example of XML-MDX Query

Then, a new XQ-Cube is returned as a result, which has the axis dimensions specified in the SELECT clause. The axis COLUMNS has two members of the dimension 'Topic': 'XML' and 'OLAP'. The axis ROWS has two members of the dimension 'Inventor': 'university' and 'industry'.

## 6 Conclusions

In this paper, we proposed XML-OLAP as a new framework for multidimensional analysis of XML warehouses. We assumed that both fact and dimension data are all represented as XML documents in XML warehouses. We proposed to construct a new type of cube named XQ-Cube from XML warehouses. An XQ-Cube is constructed from the measure data specified by an XQuery expression. We used text mining operations for the aggregation of text measure data. We proposed XML-MDX as a new multidimensional expression language for XQ-Cubes. We demonstrated its effectiveness through the example of U.S. Patent XML Warehouse. We believe our framework will contribute to the effective analysis of the vast amount of XML documents on the Web.

## Acknowledgement

This work was supported by the Post-doctoral Fellowship Program of Korea Science & Engineering Foundation (KOSEF).

## References

1. A. Abello, J. Samos and F. Saltor "Understanding Facts in a Multidimensional Object-Oriented Model," In *Proc. The 4th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP01)*, pp. 32–39, Atlanta, 2001.
2. M. Gofarelli, S. Rizzi, and B. Vrdoljak, "Data Warehouse Design from XML Sources," In *Proc. The 4th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP01)*, pp. 40–47, Atlanta, 2001.
3. W. Hümmer, A. Bauer, and G. Harde, "XCube – XML For Data Warehouses," In *Proc. The 6th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP03)*, pp. 33–40, New Orleans, Louisiana, 2003.
4. M. R. Jensen, T. H. Møller and T. B. Pedersen, "Specifying OLAP Cubes on XML Data," *Journal of Intelligent Information Systems*, Vol. 17, No. 2/3, pp. 255–280, 2001.
5. M. R. Jensen, T. H. Møller and T. B. Pedersen, "Converting XML Data To UML Diagrams For Conceptual Data Integration," In *Proc. The 1st Intl Workshop on Data Integration Over The Web*, pp. 17–31, 2001.
6. S. Lujan-Mora, J. Trujillo and P. Vassiliadis, "Advantages of UML for Multidimensional Modeling," In *Proc. the 6th Intl Conf. on Enterprise Information Systems (ICEIS 2004)*, pp. 298–305, ICEIS Press, Porto (Portugal), 2004.
7. V. Nassis, R. Rajugan, T. S. Dillon and W. Rahayu, "Conceptual Design of XML Document Warehouses," In *Proc. Data Warehousing and Knowledge Discovery, 6th International Conference, DaWaK 2004*, pp. 1–14, Zaragoza, Spain, 2004.

8. T. Niemi, M. Niinimaki, J. Nummenmaa and P. Thanisch, "Constructing an OLAP Cube from Distributed XML Data," In *Proc. The 5th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP02)*, pp. 22–27, McLean, 2002.
9. J. Pokorny, "Modelling Stars Using XML," In *Proc. The 4th ACM Intl Workshop on Data Warehousing and OLAP (DOLAP01)*, pp. 24–31, Atlanta, 2001.
10. L. I. Rusu, W. Rahayu and D. Taniar, "On Building XML Data Warehouses," In *Proc. Intelligent Data Engineering and Automated Learning - IDEAL 2004, 5th International Conference*, pp. 293–299, Exeter, UK, 2004.
11. G. Spofford, *MDX Solutions with Microsoft SQL Server Analysis Services*, John Wiley & Sons, 2001.
12. United States Patent and Trademark Office, <http://www.uspto.gov/>



# Graph-Based Modeling of ETL Activities with Multi-level Transformations and Updates

Alkis Simitsis<sup>1</sup>, Panos Vassiliadis<sup>2</sup>, Manolis Terrovitis<sup>1</sup>, and Spiros Skiadopoulos<sup>1</sup>

<sup>1</sup> National Technical University of Athens,  
Dept. of Electrical and Computer Eng., Athens, Hellas  
{asimi, mter, spiros}@dbnet.ece.ntua.gr  
<sup>2</sup> University of Ioannina, Dept. of Computer Science, Ioannina, Hellas  
pvassil@cs.uoi.gr

**Abstract.** Extract-Transform-Load (ETL) workflows are data centric workflows responsible for transferring, cleaning, and loading data from their respective sources to the warehouse. In this paper, we build upon existing graph-based modeling techniques that treat ETL workflows as graphs by (a) extending the activity semantics to incorporate negation, aggregation and self-joins, (b) complementing querying semantics with insertions, deletions and updates, and (c) transforming the graph to allow zoom-in/out at multiple levels of abstraction (i.e., passing from the detailed description of the graph at the attribute level to more compact variants involving programs, relations and queries and vice-versa).

## 1 Introduction

Conceptual and logical modeling of the design of data warehouse back-stage activities has been a relatively novel issue in the research community [3, 5, 6, 7]. The data warehouse back-stage activities are mainly implemented through tools, known as Extraction-Transformation-Loading (ETL) tools that employ data centric workflows to extract data from the sources, clean them from logical or syntactical inconsistencies, transform them into the format of the data warehouse, and eventually load these data into the warehouse.

The main issues concerning the modeling of these activities have to do (a) with the semantics of the involved activities and (b) with the exploitation of the deduced model to obtain a better understanding and a clearer evaluation of the quality of the produced design for a data warehouse scenario.

Several works in the area [2, 4] present systems tailored for ETL tasks (see also [9] for a broader discussion); nevertheless, the main focus of these works is on achieving functionality, rather than on modeling the internals or dealing with the software design or maintenance of these tasks. [3, 5] employ UML as a conceptual modeling language, whereas [7] introduces a generic graphical notation for the same task. Being defined at the conceptual level, these efforts lack a full model of the semantics of ETL workflows and a mechanism to allow the designer to navigate efficiently through large scale designs without being overwhelmed by their inherent complexity.

In our previous research, we have presented a first attempt towards a graph-based model for the definition of the ETL scenarios [6]. The model of [6, 8] treats an ETL scenario as a graph, which we call the *Architecture Graph*. Activities and data stores are modeled as the nodes of the graph; the attributes that constitute them are modeled as nodes too. Activities have input and output schemata and *provider relationships* relate inputs and outputs between data providers and data consumers. In this paper, we extend previous work in several ways. First, we complement the existing graph-based modeling of ETL activities by adding graph constructs to capture the semantics of insertions, deletions and updates. Second, we extend the previous results by adding negation, aggregation and self-joins in the expressive power of our graph-based approach. More importantly, we introduce a systematic way of transforming the Architecture Graph to allow zooming in and out at multiple levels of abstraction (i.e., passing from the detailed description of the graph at the attribute level to more compact variants involving programs, relations and queries and vice-versa). The visualization of the Architecture graph at multiple levels of granularity allows the easier understanding of the overall structure of the involved scenario, especially as the scale of the scenarios grows.

This paper is organized as follows. In Section 2, we discuss extensions to the graph model for ETL activities. Section 3 introduces a principled approach for zooming in and out the graph. In Section 4, we conclude our results and provide insights for future work.

## 2 Modeling of Side-Effects and Special Cases for ETL Activities

The purpose of this section is to present a formal logical model for the activities of an ETL environment and the extensions to existing work that we make. First, we start with the background constructs of the model, already introduced in [6, 8] and then, we move on to extend this modeling with update semantics, negations, aggregation and self-joins. We employ LDL++ [1, 10] in order to describe the semantics of an ETL scenario in a declarative nature and understandable way. LDL++ is a logic-programming, declarative language that supports recursion, complex objects and negation. Moreover, LDL++ supports external functions, choice, (user-defined) aggregation and updates.

### 2.1 Preliminaries

In this subsection, we introduce the formal model of data types, data stores and functions, before proceeding to the model of ETL activities. To this end, we reuse the modeling constructs of [6, 8] upon which we subsequently proceed to build our contribution. The basic components of this modeling framework are:

- *Data types*. Each data type  $\mathbb{T}$  is characterized by a name and a domain, i.e., a countable set of values. The values of the domains are also referred to as *constants*.
- *Attributes*. Attributes are characterized by their name and data type. For single-valued attributes, the domain of an attribute is a subset of the domain of its data

type, whereas for set-valued, their domain is a subset of the powerset of the domain of their data type  $2^{\text{dom}(T)}$ .

- A *Schema* is a finite list of attributes. Each entity that is characterized by one or more schemata will be called *Structured Entity*.
- *Records & RecordSets*. We define a *record* as the instantiation of a schema to a list of values belonging to the domains of the respective schema attributes. Formally, a recordset is characterized by its name, its (logical) schema and its (physical) extension (i.e., a finite set of records under the recordset schema). In the rest of this paper, we will mainly deal with the two most popular types of recordsets, namely *relational tables* and *record files*.
- *Functions*. A *Function Type* comprises a name, a finite list of *parameter data types*, and a single *return data type*.
- *Elementary Activities*. In the framework of [8], activities are logical abstractions representing parts, or full modules of code. An *Elementary Activity* (simply referred to as *Activity* from now on) is formally described by the following elements:
  - *Name*: a unique identifier for the activity.
  - *Input Schemata*: a finite list of one or more input schemata that receive data from the data providers of the activity.
  - *Output Schemata*: a finite list of one or more output schemata that describe the placeholders for the rows that pass the checks and transformations performed by the elementary activity.
  - *Operational Semantics*: a program, in LDL++, describing the content passing from the input schemata towards the output schemata. For example, the operational semantics can describe the content that the activity reads from a data provider through an input schema, the operation performed on these rows before they arrive to an output schema and an implicit mapping between the attributes of the input schema(ta) and the respective attributes of the output schema(ta).
  - *Execution priority*. In the context of a scenario, an activity instance must have a priority of execution, determining when the activity will be initiated.
- *Provider relationships*. These are 1:N relationships that involve attributes with a provider-consumer relationship. The flow of data from the data sources towards the data warehouse is performed through the composition of activities in a larger scenario. In this context, the input for an activity can be either a persistent data store, or another activity. Provider relationships capture the mapping between the attributes of the schemata of the involved entities. Note that a consumer attribute can also be populated by a constant, in certain cases.
- *Part\_of relationships*. These relationships involve attributes and parameters and relate them to their respective activity, recordset or function to which they belong.

The previous constructs, can be complemented by incorporating the semantics of ETL workflow in our framework. Due to the lack of space, we do not elaborate in detail on the full mechanism of the mapping of LDL rules to the Architecture Graph (including details on intra-activity and inter-activity programs); we refer the interested reader to [9] for this task. Instead, in this paper, we focus on the parts concerning

side-effect programs (which are most common in ETL environments), along with the modeling of aggregation and negation. To this end, we first need to introduce *programs* as another modeling construct.

- *Programs*. We assume that the semantics of each activity is given by a declarative program expressed in LDL++. Each program is a finite list of LDL++ rules. Each rule is identified by an (internal) rule identifier. We assume a normal form for the LDL++ rules that we employ. In our setting, there are three types of programs, and normal forms, respectively:
  - (i) *intra-activity* programs that characterize the internals of activities (e.g., a program that declares that the activity reads data from the input schema, checks for NULL values and populates the output schema only with records having non-NULL values)
  - (ii) *inter-activity* programs that link the input/output of an activity to a data provider/consumer
  - (iii) *side-effect* programs that characterize whether the provision of data is an insert, update, or delete action.

We assume that each activity is defined in isolation. In other words, the inter-activity program for each activity is a stand-alone program, assuming the input schemata of the activity as its EDB predicates. Then, activities are plugged in the overall scenario that consists of inter-activity and side-effect rules and an overall *scenario program* can be obtained from this combination.

**Side-effect programs.** We employ side-effect rules to capture database updates. We will use the generic term *database updates* to refer to insertions, deletions and updates of the database content (in the regular relational sense). In LDL++, there is an easy way to define database updates. An update expression is of the form

```
head <- query part, update part
```

and has the following semantics: (a) we make a query to the database and specify the tuples that abide by the query part and (b) we update the predicate of the update part as specified in the rule.

```
raise1(Name, Sal, NewSal) <-
  employee(Name, Sal), Sal = 1100,           (a)
  NewSal = Sal * 1.1,                       (b)
  - employee(Name, Sal),                   (c)
  + employee(Name, NewSal).                (d)
```

**Fig. 1.** Exemplary LDL++ rule for side-effect updates

For example, consider the rule depicted in Fig. 1. In Line (a) of the rule, we mark the employee tuples with salary equal to 1100 in the relation `employee(Name, Sal)`. For each the above marked tuples, Line (b) computes an updated salary with a 10% raise through the variable `NewSal`. In Line (c), we delete the originally marked tuples from the relation. Finally, Line (d) inserts the updated tuples, containing the new

salaries in the relation. In LDL updates, the order of the individual atoms is important and the query part should always advance the update part, to avoid having undesired effects from a predicate failing after an update (more details for the syntax of LDL can be found in [10]).

### 2.2 Mapping Side-Effect Programs to the Architecture Graph of a Scenario

Concerning our modeling effort, the main part of our approach lies in mapping declarative rules, expressing the semantics of activities in LDL, to a graph, which we call the Architecture Graph. In our previous work, the focus of [8] is on the input-output role of the activities instead of their internal operation. It is quite straightforward to complement this modeling with the graph of intra- and inter-activity rules [9]. In principle, activities comprise input and output schemata. Intra-activity programs and their variables facilitate the mapping of inputs to outputs. All attributes, activities and relations are nodes of the graph, connected through the proper part-of relationships. Each LDL rule connecting inputs (body of the rule) to outputs (head of the rule) is practically mapped to a set of provider edges, connecting inputs to outputs. Special purpose regulatory edges, capturing filters or joins are also part of the graph.

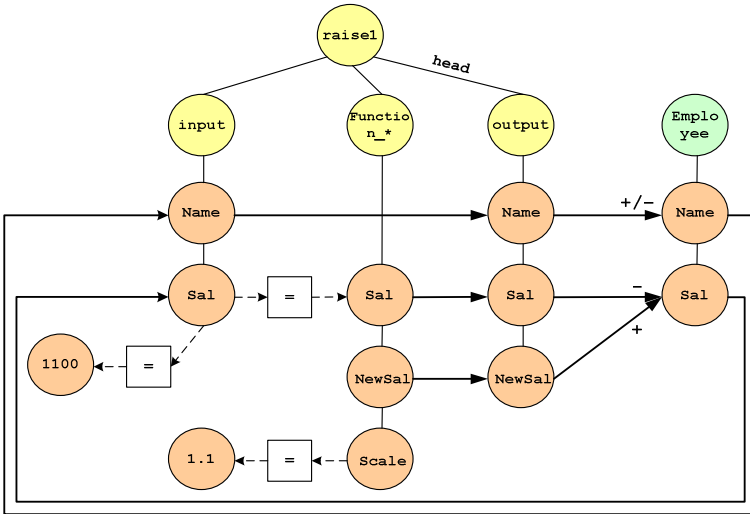


Fig. 2. Side-effects over the LDL++ rule of Fig. 1

While intra- and inter-activity rules are straightforwardly mapped to graph-based constructs, side-effects involve a rather complicated modeling, since there are both values to be inserted or deleted along with the rest of the values of a recordset. Still, there is a principled way to map LDL side-effects to the Architecture Graph.

1. A side-effect rule is treated as an activity, with the corresponding node. The output schema of the activity is derived from the structure of the predicate of the head of the rule.
2. For every predicate with a + or – in the body of the rule, a respective provider edge from the output schema of the side-effect activity is assumed. A basic syntactic restriction here is that the updated values appear in the output schema. All provider relations from the output schema to the recordset are tagged with a + or –.
3. For every predicate that appears in the rule without a + or – tag, we assume the respective input schema. Provider edges from this predicate towards these schemata are added as usual. The same applies for the attributes of the input and output schemata of the side effect activity. An obvious syntactic restriction is that all predicates appearing in the body of the rule involve recordsets or activity schemata (and not some intermediate rule).

Notice that it is permitted to have cycles in the graph, due to the existence of a recordset in the body of a rule both tagged and untagged (i.e., both with its old and new values). The old values are mapped to the input schema and the new to the output schema of the side-effect activity.

In Fig. 2, we depict an example for the usage of side-effects over the LDL++ rule of Fig. 1. Observe that `Name` is tagged both as + or –, due to its presence at two predicates, one removing the old value of `Sal` and another inserting `NewSal`, respectively. Observe, also, how the input is derived from the predicate `employee` at the body of the rule.

### 2.3 Special Cases for the Modeling of the Graph

In this subsection, we extend our basic modeling to cover special cases such as aliases, negation, aggregation and functions.

**Alias relationships.** An alias relationship is introduced whenever the same predicate appears in the same rule (e.g., in the case of a self-join). All the nodes representing these occurrences of the same predicate are connected through alias relationships to denote their semantic interrelationship. Note that due to the fact that intra-activity programs do not directly interact with external recordsets or activities, this practically involves the rare case of internal intermediate rules.

**Negation.** When a predicates appears negated in a rule body, then the respective part-of edge between the rule and the literal’s node is tagged with ‘–’. Note that negated predicates can appear only in the rule body.

**Aggregation.** Another interesting feature is the possibility of employing aggregation. In LDL, aggregation can be coded in two steps: (a) grouping of values to a bag and (b) application of an aggregate function over the values of the bag. Observe the example of Fig. 3, where data from the table `DW.PARTSUPP` are summarized, through activity `Aggregate1` to provide the minimum daily cost in view `V1`. In Fig. 3 we list the LDL program for this activity. Rules (R16–R18) explain how the data of table `DW.PARTSUPP` are aggregated to produce the minimum cost per supplier and day.

Observe how LDL models aggregation in rule R17. Then, rule R19 populates view V1 as an inter-activity program.

The graph of an LDL rule is created as usual with only 3 differences:

1. Relations which create a set from the values of a field employ a pair of regulator edges through an intermediate node '<>'.
2. Provider relations for attributes used as groupers are tagged with 'g'.
3. One of the attributes of the `aggr` function node consumes data from a constant that indicates which aggregate function should be used (e.g., `avg`, `min`, `max`).

```

R16: aggregate1.a_in(skey, suppkey, date, qty, cost) <-
      dw.partsupp(skey, suppkey, date, qty, cost)
R17: temp(skey, day, <cost>) <-
      aggregate1.a_in(skey, suppkey, date, qty, cost) .
R18: aggregate1.a_out(skey, day, min_cost) <-
      temp(skey, day, all_costs) ,
      aggr(min, all_costs, min_cost) .
R19: v1(skey, day, min_cost) <-
      aggregate1.a_out(skey, day, min_cost) .
  
```

**Fig. 3.** LDL Specification for an activity involving aggregation

**Functions.** Functions are treated as any other predicate in LDL, thus they appear as common nodes in the architecture graph. Nevertheless, there are certain special requirements for functions:

1. The function involves a list of parameters, the last of which is the return value of the function.
2. All function parameters referenced in the body of the rule either as homonyms with attributes, of other predicates or through equalities with such attributes, are linked through equality regulator relationships with these attributes.
3. The return value is possibly connected to the output through a provider relationship (or with some other predicate of the body, through a regulator relationship).

For example, observe Fig. 2 where a function involving the multiplication of attribute `Sal` with a constant is involved. Observe the part-of relationship of the function with its parameters and the regulator relationship with the first parameter and its populating attribute. The return value is linked to the output through a provider relationship.

### 3 Different Levels of Detail of the Architecture Graph

The Architecture Graph can become a complicated construct, involving the full detail of activities, recordsets, attributes and their interrelationships. Although it is important and necessary to track down this information at design time, in order to formally specify the scenario, it quite clear that this information overload might be cumbersome to manage at later stages of the workflow lifecycle. In other words, we need to provide the user with different versions of the scenario, each at a different level of detail.

We will frequently refer to these abstraction levels of detail simply, as *levels*. We have already defined the Architecture Graph at the *attribute level*. The attribute level is the most detailed level of abstraction of our framework. Yet, coarser levels of detail can also be defined. The *schema level*, abstracts the complexities of attribute interrelationships and presents only how the input and output schemata of activities interplay in the data flow of a scenario. In fact, due to the composite structure of the programs that characterize an activity, there are more than one variants that we can employ for this description. Finally, the coarser level of detail, the *activity level*, involves only activities and recordsets. In this case, the data flow is described only in terms of these entities.

**Architecture Graph at the Schema Level.** Let  $G_S(V_S, E_S)$  be the architecture graph of an ETL scenario at the schema level. The scenario at the schema level has schemata, functions, recordsets and activities for nodes. The edges of the graph are part-of relationships among structured entities and their corresponding schemata and provider relationships among schemata. The direction of provider edges is again from the provider towards the consumer and the direction of the part-of edges is from the container entity towards its components (in this case just the involved schemata). Edges are tagged appropriately according to their type (part-of or provider).

Intuitively, at the schema level, instead of fully stating which attribute populates another attribute, we trace only how this is performed through the appropriate schemata of the activities. A program capturing the semantics of the transformations and cleanings that take place in the activity is the means through which the input and output schemata are interconnected. If we wish, instead of including all the schemata of the activity as they are determined by the intermediate rules of the activity's program, we can present only the program as a single node of the graph, to avoid the extra complexity.

There is a straightforward way to zoom out the Architecture Graph at the attribute level and derive its variant at the schema level. For each node  $x$  of the architecture graph  $G(V, E)$  representing a schema:

1. for each provider edge  $(x_a, y)$  or  $(y, x_a)$ , involving an attribute of  $x$  and an entity  $y$ , external to  $x$ , introduce the respective provider edge between  $x$  and  $y$  (unless it already exists, of course);
2. remove the provider edges  $(x_a, y)$  and  $(y, x_a)$  of the previous step;
3. remove the nodes of the attributes of  $x$  and the respective part-of edges.

We can iterate this simple algorithm over the different levels of part-of relationships, as depicted in Fig. 4.

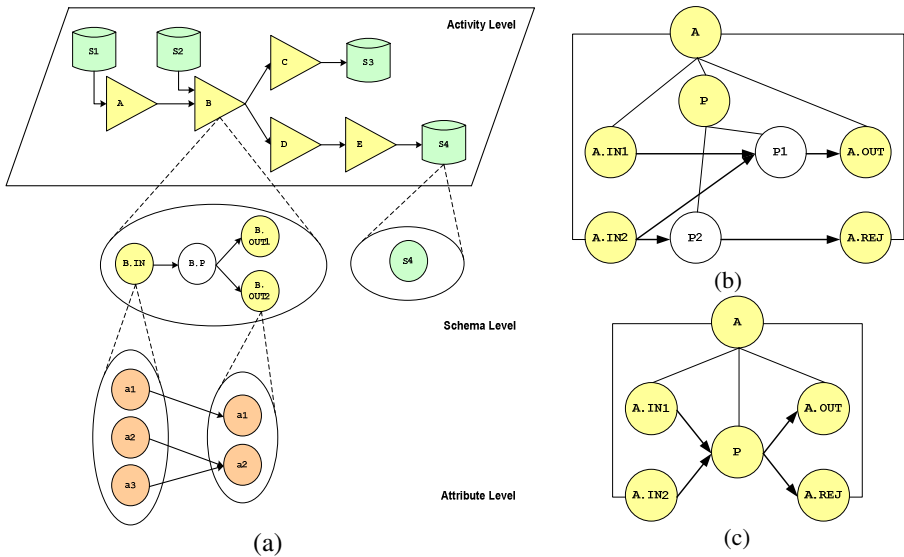
**Architecture Graph at the Activity Level.** In this paragraph, we will deal with the model of ETL scenarios as graphs at the activity level. Only activities and recordsets are part of a scenario at this level. Let  $G_A(V_A, E_A)$  be the architecture graph of an ETL scenario at the activity level. The scenario at the activity level has only recordsets and activities for nodes and a set of provider relationships among them for edges. The provider relationships are directed edges from the provider towards the consumer entity.

Intuitively, a scenario is a set of activities, deployed along a graph in an execution sequence that can be linearly serialized through topological ordering. There is a straightforward way to zoom out the Architecture Graph at the schema level and derive



its variant at the activity level. For each node  $x$  of the architecture graph  $G_A (V_A, E_A)$  representing a structured entity (i.e., activity or recordset):

1. for each provider edge  $(x_c, y)$  or  $(y, x_c)$ , involving a schema of  $x$  and an entity  $y$ , external to  $x$ , introduce the respective provider edge between  $x$  and  $y$  (unless it already exists, of course);
2. remove the provider edges  $(x_c, y)$  and  $(y, x_c)$  of the previous step;
3. remove the nodes of the schema(ta) and program (if  $x$  is an activity) of  $x$  and the respective part-of edges.



**Fig. 4.** Zooming in/out. (a) different levels of detail for ETL workflows; (b) an activity with two input schemata populating an output and a rejection schema as follows: a subprogram P1 is assigned the population of the output schema only and a subprogram P2 populates only the rejection schema using only one input schema; and (c) a single node abstracts the internal structure of the activity.

**Discussion.** Navigating through different levels of detail is a facility that primarily aims to make the life of the designer and the administrator easier throughout the full range of the lifecycle of the data warehouse. Through this mechanism, the designer can both avoid the complicated nature of parts that are not of interest at the time of the inspection and drill-down to the lowest level of detail for the parts of the design that he is interested in.

Moreover, apart from this simple observation, we can easily show how our graph-based modeling provides the fundamental platform for employing software engineering techniques for the measurement of the quality of the produced design [9]. Zooming in and out the graph in a principled way allows the evaluation of the overall design both at different depth of granularity and at any desired breadth of range (i.e., by isolating only the parts of the design that are currently of interest).

## 4 Conclusions

Previous research in the logical modeling of ETL workflows has identified graph-based techniques that capture the high-level structure of these workflows. In this paper, we have extended the semantics of the involved ETL activities to incorporate negation, aggregation and self-joins. Moreover, we have complemented this semantics in order to handle insertions, deletions and updates. Finally, we have provided a principled method for transforming the architecture graph of an ETL scenario to allow zoom-in/out at multiple levels of abstraction. This way, we can move from the detailed description of the graph at the attribute level to more compact variants involving programs, relations and queries and vice-versa.

Research can be continued in more than one direction, e.g., towards the derivation of precise algorithms for the evaluation of the impact of changes in the Architecture Graph. Finally, a field-study of the usage of the Architecture Graph in all the phases of a data warehouse project can also be pursued.

## Acknowledgments

This work is co-funded by the European Social Fund (75%) and National Resources (25%) - Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS.

## References

- [1] S. Ceri, G. Gottlob, L. Tanca. Logic Programming and Databases. Springer-Verlag, 1990.
- [2] H. Galhardas, D. Florescu, D. Shasha and E. Simon. Ajax: An Extensible Data Cleaning Tool. In Proc. of ACM SIGMOD'00, pp. 590, Dallas, Texas, 2000.
- [3] S. Lujan-Mora, P. Vassiliadis, J. Trujillo. Data Mapping Diagrams for Data Warehouse Design with UML. In Proc. 23rd International Conference on Conceptual Modeling (ER 2004), pp. 191-204, Shanghai, China, 2004.
- [4] V. Raman, J. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In Proc. of VLDB'01, pp. 381-390, Roma, Italy, 2001.
- [5] J. Trujillo, S. Luján-Mora: A UML Based Approach for Modeling ETL Processes in Data Warehouses. In Proc. of ER'03, pp. 307-320, Chicago, USA, 2003.
- [6] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Modeling ETL Activities as Graphs. In Proc. of DMDW'02, pp. 52-61, Toronto, Canada, 2002.
- [7] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Conceptual Modeling for ETL Processes. In Proc. of DOLAP'02, pp. 14-21, McLean, Virginia, USA, 2002.
- [8] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis. A Framework for the Design of ETL Scenarios. In Proc. of CAiSE'03, pp. 520-535, Austria, 2003.
- [9] P. Vassiliadis, A. Simitsis, M. Terrovitis, S. Skiadopoulos. Blueprints for ETL workflows (long version). Available through [http://www.cs.uoi.gr/~pvassil/publications/2005\\_ER\\_AG/ETL\\_blueprints\\_long.pdf](http://www.cs.uoi.gr/~pvassil/publications/2005_ER_AG/ETL_blueprints_long.pdf)
- [10] C. Zaniolo. LDL++ Tutorial. UCLA. <http://pike.cs.ucla.edu/ldl/>, December 1998.

# Extending UML 2 Activity Diagrams with Business Intelligence Objects\*

Veronika Stefanov, Beate List, and Birgit Korherr

Women's Postgraduate College for Internet Technologies,  
Institute of Software Technology and Interactive Systems,  
Vienna University of Technology  
{lastname}@wit.tuwien.ac.at  
<http://wit.tuwien.ac.at>

**Abstract.** Data Warehouse (DWH) information is accessed by business processes. Today, no conceptual models exist that make the relationship between the DWH and the business processes transparent. In this paper, we extend a business process modeling diagram, namely the UML 2 activity diagram with a UML profile, which allows to make this relationship explicit. The model is tested with example business processes.

## 1 Introduction

A Data Warehouse (DWH) is more than just another big database. It is defined as “a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management’s decision-making process“ [7]. In modern organisations, DWHs play a crucial role, as more and more business processes require information from the DWH. A business process is “a group of tasks that together create a result of value to a customer” [6], and describes how work is done within an organization. When a person applies for a loan in a bank for example, the DWH is an integral part of the loan application business process. The applicant is scrutinized to find out if she or he has caused a financial loss previously, or has changed identity and caused damage under a different name. The business processes of designing new products in a telecommunication company or an airline, or composing the product range of a supermarket for example, requires comprehensive information on the customer behavior covered by the DWH. There are lots of examples showing how important DWHs have become for business processes.

Surprisingly, this knowledge – how dynamic business structures interact with the DWH and how the DWH is being used in every day business life – is not made explicit in existing models. There is a need for an integrated model of processes and DWHs to make the relationship between the DWH and the business processes more transparent. To bridge this gap, we extend a business process modeling diagram,

---

\* This research has been funded by the Austrian Federal Ministry for Education, Science, and Culture, and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

namely the UML 2 activity diagram [12], with a *UML profile for Business Intelligence (BI) Objects*, to be able to create models that show

- where and how business processes use a DWH environment, and
- which parts of the business processes depend on which parts of the DWH.

UML profiles provide an extension mechanism for building UML models for particular domains or purposes [12]. We utilize this extension mechanism, because the UML Profile for BI Objects provides the advantage that DWH people are able to view business process models and the interaction with a DWH in a well-known notation. In addition to the reuse of the UML notation, these models can be easily presented and edited with UML tools, as almost all UML tools support UML profiles.

We use the term Business Intelligence (BI) instead of DWH, as it represents a broader approach to decision support data. We see BI as all kinds of applications and technologies for storing, analyzing, and accessing data to help enterprises to make better business decisions. BI objects cover a broad range of object types. We distinguish between data repositories (DWHs, data marts, or operational data stores), data objects representing data models of data repositories (entities or facts), and presentation objects representing tools (reports or analysis tools). These BI objects can be accessed by the activities of a business process, or in this case by the actions of the UML 2 activity diagram.

The contribution of the UML profile for BI Objects is:

- The model provides the bigger picture to DWH designers, as it shows how the DWH and other BI objects are accessed by business processes.
- The model links static BI structures and dynamic business structures.
- The UML Profile provides BI objects on different aggregation levels and thus enables the modeler to choose the right level of detail for different purposes or target audiences. The modeler may model a high level data repository access of a business process, e.g. the access of a data mart or DWH, or describe the access at a more detailed level, e.g. the access of a certain fact or entity. Furthermore, modelers may also show the access of an analysis tool.
- The model can support the design phase of a BI project, by making it possible to describe the business requirements for the DWHs or data marts in a ‘to be business process model’. The DWH department can then prioritize the projects accordingly.
- By relating DWHs or data marts to decisions in business processes, such a model can be used to justify the costs of BI projects.
- The model can also be used to support estimates of the cost of usage, as well as for risk management: if the data quality in a certain area is bad, a data mart fails or data is corrupted, an integrated model enables better reactions because it is known which business processes will be affected.
- Finally, the model also allows to discover parts of the DWH or data mart data model which are not accessed at all, permitting the DWH department to decide if these parts should be further maintained.

Based on the meta-model in Section 2, we have developed a UML Profile for BI Objects extending UML 2 activity diagrams in Section 3. The UML profile is tested by example business processes in Section 4. Section 5 covers related work.

## 2 Meta-model of Business Intelligence Objects

We extend the UML 2 activity diagram with a UML Profile for BI Objects to enable the creation of models that integrate information about where a business process makes use of data for decision support. These models make the otherwise hidden knowledge about the relationships between the business processes and BI explicit. This section describes the meta-model of BI Objects.

What is a BI object? We have identified three main categories of BI objects: *Data Repositories* (representing the elements of the DWH architecture), *Data Objects* (representing the data model of a certain repository), and *Presentation Objects* (representing the means of presentation, either a static report or an interactive analysis). The relationships between the BI objects are shown in Fig. 1.

BI objects chosen for a model depend on the target audience and the level of detail of the model. In an overview business process model suited for DWH managers, one might show the DWH or individual data marts as a whole. In a more detailed model for developers, sub-processes can be described as accessing individual entities and facts. Additionally, decision makers often receive relevant data in form of reports, for instance a report on sales data for the past fiscal year, which may also be relevant for business process modeling.

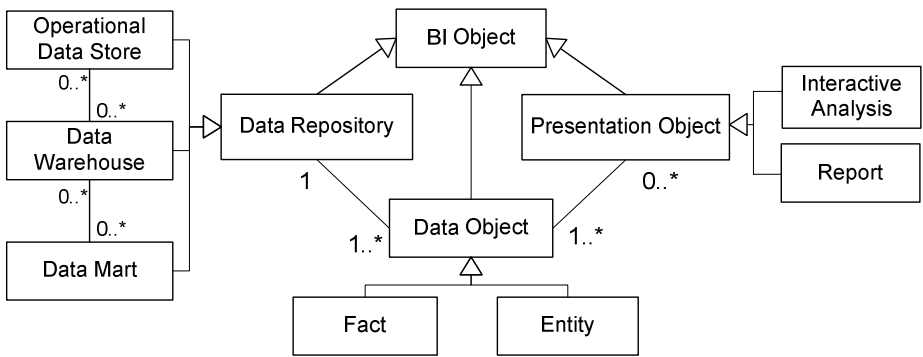


Fig. 1. Meta-model of Business Intelligence Objects

### 2.1 Data Repositories

*Data Repositories* are the first type of BI object that can be modeled in relation to a business process. They basically represent different types of databases as used in DWH settings. The types of data repositories occurring in a given situation depend on

the DWH architecture in an organization. Also, several different data repositories may exist in parallel. Our approach is not limited to any specific DWH architecture, but can be applied to a wide selection of architecture types. In order to allow the greatest possible flexibility and provide meaningful content in the models, we have identified three basic types of data repositories: the *Data Warehouse (DWH)*, the *Data Mart* and the *Operational Data Store (ODS)*.

Depending on the architecture, different combinations of BI data repositories may occur in an organization. In large multinational organizations it is not uncommon to have more than one DWH. Within an organization a large DWH often co-exists with smaller data marts, departmental subsets of a DWH focused on selected subjects [2]. The data mart might be based on the DWH, obtaining its data from there, and acting as a kind of materialized view on the DWH. In another case, each data mart may be created individually by a department without an underlying DWH. To make operations spanning several data marts possible, they may later be integrated into an organization-wide DWH. Also, there may be none, one or more ODS, located between the operational systems and the DWH [5]. Depending on the architecture, end user applications may query individual data marts and/or the DWH, or even access the data in the ODS directly.

## 2.2 Data Objects

In order to provide a more detailed view on the data, we also want to model the individual data entities contained in the data repositories. These *Data Objects* are generally represented in conceptual data models. For example, if a business process needs data on the revenue of a certain product range, it can be modeled to access the corresponding data object directly. In BI settings, there are two common types of data models: entity-relationship (E/R) models [3] and multidimensional models [2][4][9]. Which model is used depends on the type of repository, the overall architecture, and the preferences of the designers. The data objects of an E/R model that can be accessed by an activity of a business process are *Entities*. In the case of the multidimensional model, they are *Facts*.

## 2.3 Presentation Objects

In an organization employing BI techniques, there are usually tools and applications providing users with prepackaged information that has been compiled for them. We call these collections of information *Presentation Objects*, and have identified two different types: *Report* or *Interactive Analysis*. A report displays a predefined set of queries, for example a report on sales in the south region for the 4<sup>th</sup> quarter of 2004. The values contained in a report do not change over time. An interactive analysis is a tool, e.g. an OLAP tool. In this case, the queries or analysis operations are not predefined but can be chosen by the user. The values are regularly updated and can be used for continuous performance monitoring. In a business process model we can for instance show a certain report that is accessed by an activity.

### 3 The UML Profile for Business Intelligence Objects

UML offers a possibility to extend and adapt its meta-model to a specific area of application through the creation of profiles. UML profiles are UML packages with the stereotype «profile». A profile can extend a meta-model or another profile [12] while preserving the syntax and semantic of existing UML elements. It adds elements which extend existing classes. UML profiles consist of *stereotypes*, *constraints* and *tagged values*.

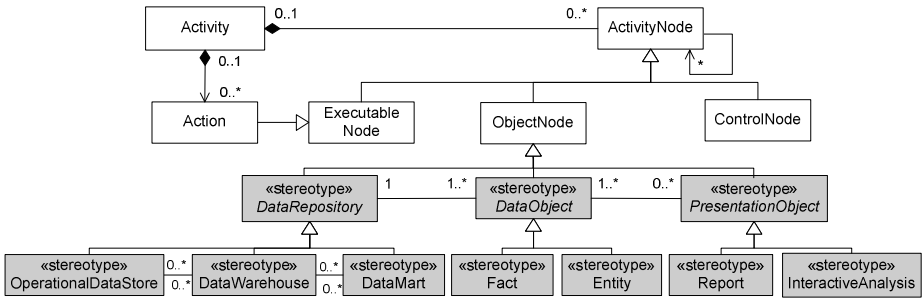


Fig. 2. Extending the UML2 Meta-Model with Stereotypes for BI Objects

A stereotype is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually meta-classes from the UML meta-model, for instance the meta-class «Class», but can also be stereotypes from another profile. A stereotype can have its own notation, e.g. a special icon.

Constraints are applied to stereotypes in order to indicate restrictions. They specify pre- or post conditions, invariants, etc., and must comply with the restrictions of the base class [12]. Constraints can be expressed in any language, such as programming languages or natural language. We use the Object Constraint Language (OCL) [11] in our profile, as it is more precise than natural language or pseudocode, and widely used in UML profiles.

Tagged values are additional meta-attributes assigned to a stereotype, specified as name-value pairs. They have a name and a type and can be used to attach arbitrary information to model elements.




We extend the UML 2 activity diagram with a UML Profile for BI Objects, creating an integrated model of processes and BI objects to make the relationship between the DWH environment and the business processes more transparent. Activity diagrams are used in UML for modeling processes, workflows, and computations. In Fig. 2 we show a part of the UML 2 meta-model related to activity diagrams (light) to illustrate how the stereotypes we designed (dark) fit into to the existing meta-model.

In an UML 2 activity diagram, a single activity, representing a process or part of a process, is modeled. An activity may include any number of activity nodes, such as individual actions, control nodes (e.g. splits and joins), and object nodes. These nodes can be arranged to form sequential or concurrent processes, and several activity diagrams can be connected to describe larger processes.

In the UML Profile for BI Objects, we use the class *Object Node* as base class for all stereotypes. The OMG has defined an object node as an “activity node that indicates an instance of a particular classifier, possibly in a particular state, may be available at a particular point in the activity” [12]. Therefore, object nodes represent concrete instances of information objects, which are input or output parameters of an activity. They are suited for the purpose of showing when a (sub-)process accesses a BI object, as the BI objects amount to input parameters of activities.

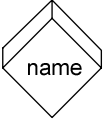

As described in the meta-model in Section 2, BI objects can be classified into three larger types. We therefore have defined three abstract top-level stereotypes, «DataRepository», «DataObject», and «PresentationObject». The stereotypes «DataWarehouse», «OperationalDataStore» and «DataMart» are derived from «DataRepository». Their specifications are listed in Table 1. The stereotype «DataObject» can be further specialized into «Fact» and «Entity», as shown in Table 2. Finally, the stereotypes «Report» and «InteractiveAnalysis» are specializations of «PresentationObject», as listed in Table 3. The semantics of the individual elements were described in greater detail in Section 2.

**Table 1.** Data Repositories: Specification of Stereotypes



Name	<b>DataRepository</b>	
Base Class	ObjectNode	
Description	A data repository represents a type of database used in data warehouse environments. The stereotypes DataWarehouse, DataMart, and OperationalDataStore are derived from DataRepository.	
Constraints	A DataRepository must be related to at least one DataObject: context DataRepository inv: Self.dataObject->size() >= 1	
Tagged Values	isMultidimensional <ul style="list-style-type: none"> <li>• Type: UML::Datatypes::Boolean</li> <li>• Multiplicity: 1</li> <li>• Description: Indicates whether the data model of the DataRepository is a multidimensional data model</li> </ul>	
Name	<b>DataWarehouse</b>	<p style="text-align: center;"><b>Notation</b></p> 
Base Class	DataRepository	
Description	A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management’s decision-making process [7].	
Tagged Values	None	
Constraints	None	
Name	<b>DataMart</b>	<p style="text-align: center;"><b>Notation</b></p> 
Base Class	DataRepository	
Description	A data mart is a departmental subset of a DWH focused on a single subject area [2].	
Tagged Values	None	
Constraints	None	
Name	<b>OperationalDataStore</b>	<p style="text-align: center;"><b>Notation</b></p> 
Base Class	DataRepository	
Description	An operational data store is located between the operational systems and the DWH [5].	
Tagged Values	None	
Constraints	None	



**Table 2.** Data Objects: Specification of Stereotypes

Name	<b>DataObject</b>	
Base Class	ObjectNode	
Description	A data object is part of the data model contained in a data repository. The stereotypes Fact and Entity are derived from DataObject.	
Tagged Values	None	
Constraints	A DataObject must belong to exactly one DataRepository: context DataObject inv: self.dataRepository.size() = 1	
	The corresponding class must have at least one attribute: context DataObject inv: self.type.allAttributes()->size() >= 1	
Name	<b>Fact</b>	<b>Notation</b> 
Base Class	DataObject	
Description	A fact is a data object of a multidimensional data model.	
Tagged Values	None	
Constraints	The DataRepository containing a fact must have a multidimensional data model: context Fact inv: self.isType(Fact) implies self.dataRepository.isMultidimensional	
Name	<b>Entity</b>	<b>Notation</b> 
Base Class	DataObject	
Description	An entity is a data object of an E/R model.	
Tagged Values	None	
Constraints	The DataRepository containing an entity must not have a multidimensional data model: context Entity inv: self.isType(Entity) implies not self.dataRepository.isMultidimensional	

**Table 3.** Presentation Objects: Specification of Stereotypes

Name	<b>PresentationObject</b>	
Base Class	ObjectNode	
Description	A presentation object is a document or tool used to present information to a user. The stereotypes Report and InteractiveAnalysis are derived from PresentationObject.	
Tagged Values	None	
Constraints	A PresentationObject must have at least one DataObject: context PresentationObject inv: self.dataObject->size() >= 1	
Name	<b>Report</b>	<b>Notation</b> 
Base Class	PresentationObject	
Description	A report displays a predefined set of queries.	
Tagged Values	None	
Constraints	None	
Name	<b>InteractiveAnalysis</b>	<b>Notation</b>  Name
Base Class	PresentationObject	
Description	An interactive analysis is a tool that allows the user to freely explore information.	
Tagged Values	None	
Constraints	None	

### 4 Examples

We present three examples that demonstrate the application of the UML Profile for BI Objects developed in Section 3, each illustrating a different aspect. The first example introduces a simple UML 2 activity diagram with BI objects, the second example illustrates how UML «selection» notes can be used in combination with BI objects to provide more detail on data access, and the third example demonstrates how a more complicated business process can be modeled on a higher level of abstraction.

The example activity diagram in Fig. 3 describes the well-known process of a passenger checking in at an airport. Two parties are involved in this activity, the passenger and the check-in desk. The process starts with the action “present documents”: the passenger presents the travel documents at the check-in desk. Two items, the ticket and the passport, are passed to the “check identity” action performed by the check-in desk. In order perform its task, the action also needs access not only to the two documents but also to the entity “reservation”. Therefore, it only starts if all three necessary inputs are available. After the identity check has concluded, the check-in desk decides on a possible upgrade. The action “decide on upgrade” needs data from the Customer Relationship Management (CRM) data mart. The data mart contains the frequent flyer status of the passenger in question. Data on the current flights situation, (e.g., whether another flight to the same destination is cancelled or overbooked, meaning that no upgrades are available) is provided by an interactive analysis tool. The “decide on upgrade” action therefore can only begin when the identity check has concluded and the two BI objects are available. It produces a boarding pass as output. The passenger can proceed to the gate as soon as he or she has received the boarding pass. Alternative paths, such as the identity check failing, were left out for sake of clarity of the example.

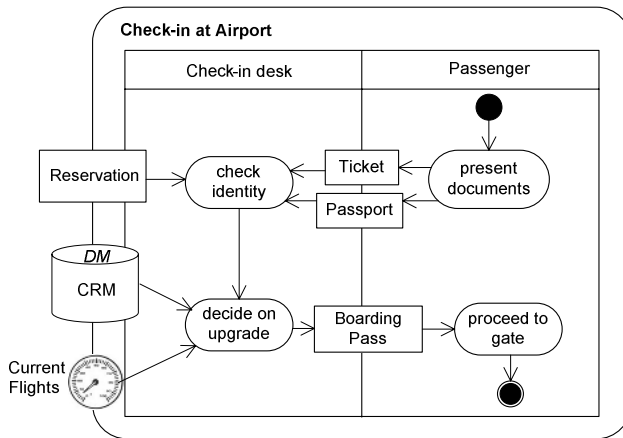


Fig. 3. Airport Check-In Business Process

A large business process can be modeled by linking together several activity diagrams, each describing a small sub-process, such as the part of the process of designing and organizing a promotion of a single product (e.g. a 30 percent discount on a brand of soap) shown in Fig. 4. In the initial step of choosing the product, a report on past promotions is analyzed in order to identify products suitable for a profitable promotion. Therefore, the action “analyze past promotions” has a set of products, e.g., those that seem promising, as output. In the following “choose product” action, a product is chosen based on how many items of the product were sold in the past (i.e. the sales information provided by the “Sales” fact) and whether enough items are on stock (i.e. inventory information from the ODS system). Only data on the products selected before should be read from the fact table and the ODS. In an activity diagram, a «selection» note attached to the object flow between an object node and an action can be used to specify selection behaviour. In the example presented here, the OCL statement checks whether a product in the BI object – the “Sales fact” or the ODS – is contained in the list of promising products.

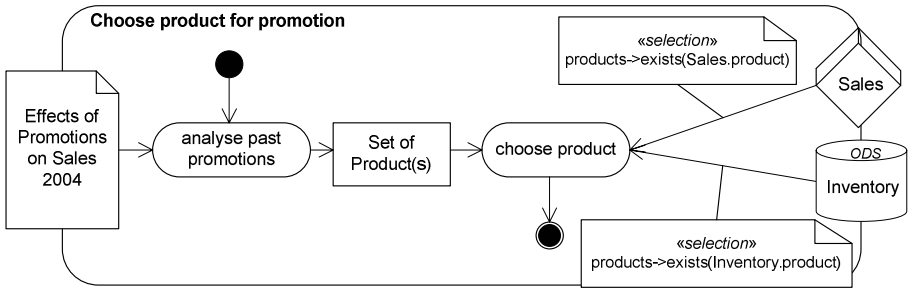


Fig. 4. Product Promotion: The Sub-Process of Choosing the Product

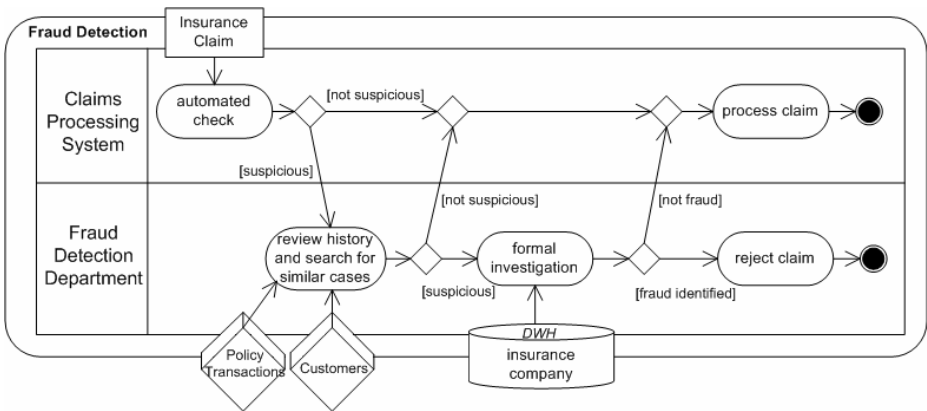


Fig. 5. Fraud Detection Business Process

During a fraud detection process at an insurance company (Fig. 5), insurance claims are subjected to a three-step analysis, aimed at recognizing all potentially fraudulent claims before they might be processed and paid. The activity “fraud detection” is started by the arrival of an insurance claim. The claim is first exposed to an extensive automated check by the claim processing system. All claims judged as being suspicious are forwarded to the fraud detection department, whereas the others are processed normally. The suspicious claims are then reviewed. In this action the results of the automated check as well as the history of the customer and the insurance policy are analyzed, to identify patterns and/or similar cases. Therefore, the action needs access to two fact tables: “Customers” and “Policy Transactions”. The claims that continue to be suspect are then formally investigated, whereas the claims re-established as genuine are returned to the claim processing system. The action “formal investigation” represents a thorough search for further clues in order to provide answers to any open questions. As the queries necessary in this step are different in every case and cannot be predicted, the action requires the whole data warehouse of the insurance company as input. All claims finally identified as fraudulent are rejected.

## 5 Related Work

There are a lot of conceptual modelling languages available for business processes or DWHs. But there are no models that focus on the relationship between these two domains. The conceptual DWH diagrams available for the different stages of the DWH process, e.g. for multidimensional models [10] or ETL processes [14], do not address the link to business processes at all. Business process diagrams that address the static structure of databases do not address the particularities of DWHs and BI.

Event-Driven Process Chains (EPC) [8] incorporate a data view, targeting operational data bases. To provide the data view with a conceptual model, Chen’s entity-relationship (ER) model was adopted, since it was the most widespread model in the area of data modelling. Today, the UML class diagram is also used. EPC functions perform read or write operations on E/R entities or UML classes. The UML Profile for BI Objects is based on a similar concept, but accounts for the particularities of DWH settings.

In UML 2 activity diagrams [12], data store nodes represent data. A UML 2 action node can perform read or write operations, comparable to the EPC function. The data store node is not necessarily linked with a UML class or database.

The Business Process Modeling Notation (BPMN) [1] provides data objects, which are used and updated during the process. The data object can be used to represent many different types of objects, both electronic or physical.

## 6 Conclusion

In this work, we have addressed the missing link in conceptual modeling between the static structures of the DWH and the dynamic structures of business processes. To bridge this gap, we have extended the UML 2 activity diagram with a UML Profile

for Business Intelligence (BI) Objects. The model shows where and how business processes use a DWH environment, and which parts of the business processes depend on which parts of the DWH. The DWH environment is specified in terms of several types of BI objects, representing the different types of data repositories, their data models and the means of presentation. These BI objects can be accessed by actions of UML 2 activity diagrams. The profile was applied to several example processes.

## References

- [1] Business Process Modeling Notation (BPMN), Specification BPMN 1.0 May 3, 2004, <http://www.bpmn.org>
- [2] S. Chaudhuri and U. Dayal, An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record, v.26 n.1, p.65-74, March 1997.
- [3] P. Chen, The Entity-Relationship Model - Toward a Unified View of Data. ACM Trans. Database Syst. 1(1): 9-36, 1976.
- [4] M. Golfarelli, D. Maio, S. Rizzi, Conceptual Design of Data Warehouses from E/R Schema, Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7, p.334, January 06-09, 1998.
- [5] M. Golfarelli, S. Rizzi, I. Cella, Beyond Data Warehousing: What's next in business intelligence? In Proceedings 7th International Workshop on Data Warehousing and OLAP (DOLAP 2004), Washington DC, 2004.
- [6] M. Hammer, Beyond Reengineering - How the process-centered organization is changing our work and our lives. Harper Collins Publishers 1996.
- [7] W. H Inmon, R. D. Hackethorn, Using the Data Warehouse. New York: John Wiley & Sons, 1994.
- [8] G. Keller, M. Nüttgens, A.-W. Scheer, Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)", In: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, Saarbrücken 1992.
- [9] R. Kimball and M. Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, second ed., John Wiley & Sons, Inc., 2002.
- [10] S. Luján-Mora, J. Trujillo, I. Song, Extending UML for Multidimensional Modeling. 5th International Conference on the Unified Modeling Language (UML 2002), p. 290-304: LNCS 2460, Springer, Dresden, 2002.
- [11] Object Management Group, Inc.: UML 2.0 OCL Specification. <http://www.omg.org/docs/ptc/03-10-14.pdf> (22/4/2005)
- [12] Object Management Group, Inc.: UML 2.0 Superstructure <http://www.omg.org/cgi-bin/apps/doc?ptc/04-10-02.zip> (22/4/2005).
- [13] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley, 2004.
- [14] J. Trujillo, S. Luján-Mora, A UML Based Approach for Modeling ETL Processes in Data Warehouses, 22nd International Conference on Conceptual Modeling (ER 2003), p. 307-320: LNCS 2813, Springer, Chicago, 2003.

# Automatic Selection of Bitmap Join Indexes in Data Warehouses

Kamel Aouiche, Jérôme Darmont, Omar Boussaïd, and Fadila Bentayeb

ERIC Laboratory – University of Lyon 2,  
5, av. Pierre Mendès-France,  
F-69676 BRON Cedex – France  
{kaouiche, jdarmont, boussaïd, bentayeb}@eric.univ-lyon2.fr

**Abstract.** The queries defined on data warehouses are complex and use several join operations that induce an expensive computational cost. This cost becomes even more prohibitive when queries access very large volumes of data. To improve response time, data warehouse administrators generally use indexing techniques such as star join indexes or bitmap join indexes. This task is nevertheless complex and fastidious. Our solution lies in the field of data warehouse auto-administration. In this framework, we propose an automatic index selection strategy. We exploit a data mining technique ; more precisely frequent itemset mining, in order to determine a set of candidate indexes from a given workload. Then, we propose several cost models allowing to create an index configuration composed by the indexes providing the best profit. These models evaluate the cost of accessing data using bitmap join indexes, and the cost of updating and storing these indexes.

## 1 Introduction

Data warehouses are generally modelled according to a star schema that contains a central, large fact table, and several dimension tables that describe the facts [10,11]. The fact table contains the keys of the dimension tables (foreign keys) and measures. A decision–support query on this model needs one or more joins between the fact table and the dimension tables. These joins induce an expensive computational cost. This cost becomes even more prohibitive when queries access very large data volumes. It is thus crucial to reduce it.

Several database techniques have been proposed to improve the computational cost of joins, such as hash join, merge join and nested loop join [14]. However, these techniques are efficient only when a join applies on two tables and data volume is relatively small. When the number of joins is greater than two, they are ordered depending on the joined tables (join order problem). Other techniques, used in the data warehouse environment, exploit join indexes to pre-compute these joins in order to ensure fast data access. Data warehouse administrators then handle the crucial task of choosing the best indexes to create (index selection problem). This problem has been studied for many years in databases [1,4,5,6,7,12,18]. However, it remains largely unresolved in data warehouses. Existing research studies may be clustered in two families: algorithms

that optimize maintenance cost [13] and algorithms that optimize query response time [2,8,9]. In both cases, optimization is realized under the constraint of the storage space. In this paper, we focus on the second family of solutions, which is relevant in our context because they aim to optimize query response time.

In addition, with the large scale usage of databases in general and data warehouses in particular, it is now very important to reduce the database administration function. The aim of auto-administrative systems is to administrate and adapt themselves automatically, without loss (or even with a gain) in performance. In this context, we proposed a method for index selection in databases based on frequent itemset extraction from a given workload [3]. In this paper, we present the follow-up of this work. Since all candidate indexes provided by the frequent itemset extraction phase cannot be built in practice due to system and storage space constraints, we propose a cost model-based strategy that selects the most advantageous indexes. Our cost models estimate the data access cost using bitmap join indexes, and their maintenance and storage cost.

We particularly focus on bitmap join indexes because they are well-adapted to data warehouses. Bitmap indexes indeed make the execution of several common operations such as **And**, **Or**, **Not** or **Count** efficient by having them operating on bitmaps, in memory, and not on the original data. Furthermore, joins are pre-computed at index creation time and not at query execution time. The storage space occupied by bitmaps is also low, especially when the indexed attribute cardinality is not high [17,19]. Such attributes are frequently used in decision-support query clauses such as **Where** and **Group by**.

The remainder of this paper is organized as follows. We first remind the principle of our index selection method based on frequent itemset mining (Section 2). Then, we detail our cost models (Section 3) and our index selection strategy (Section 4). To validate our work, we also present some experiments (Section 5). We finally conclude and provide research perspectives (Section 6).

## 2 Index Selection Method

In this section, we present an extension to our work about the index selection problem [3]. The method we propose (Figure 1) exploits the transaction log (the set of all the queries processed by the system) to recommend an index configuration improving data access time.

We first extract from a given workload a set of so called indexable attributes. Then, we build a “query-attribute” matrix whose rows represent workload queries and whose columns represent a set of all the indexable attributes. Attribute presence in a query is symbolized by one, and absence by zero. It is then exploited by the Close frequent itemset mining algorithm [16]. Each itemset is analyzed to generate a set of candidate indexes. This is achieved by exploiting the data warehouse metadata (schema: primary keys, foreign keys; statistics. . .). Finally, we prune the candidate indexes using the cost models presented in Section 3, before effectively building a pertinent index configuration. We detail these steps in the following sections.

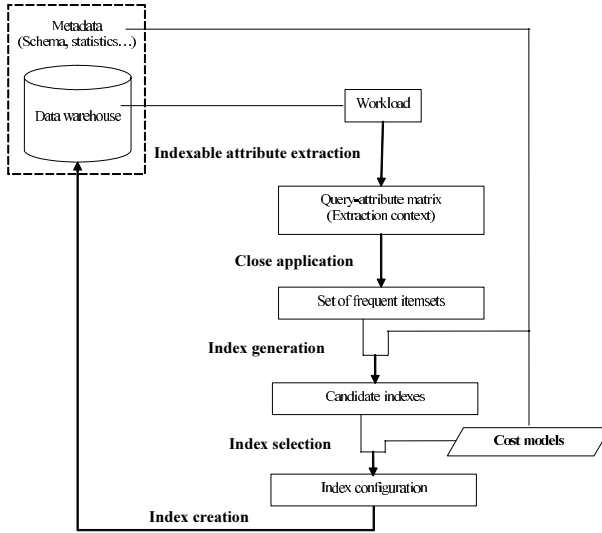


Fig. 1. Automatic index selection strategy

### 3 Cost Models

The number of candidate indexes is generally as high as the input workload is large. Thus, it is not feasible to build all the proposed indexes because of system limits (limited number of indexes per table) or storage space constraints. To circumvent these limitations, we propose cost models allowing to conserve only the most advantageous indexes. These models estimate the storage space (in bytes) occupied by bitmap join indexes, the data access cost using these indexes and their maintenance cost expressed in number of input/output operations (I/Os). Table 1 summarizes the notations used in our cost models.

#### 3.1 Bitmap Join Index Size

The space required to store a simple bitmap index linearly depends on the indexed attribute cardinality and the number of tuples in the table on which the

Table 1. Cost model parameters

Symbol	Description
$ X $	Number of tuples in table $X$ or cardinality of attribute $X$
$S_p$	Disk page size in bytes
$p_X$	Number of pages needed to store table $X$
$S_{pointer}$	Page pointer size in bytes
$m$	B-tree order
$d$	Number of bitmaps used to evaluate a given query
$w(X)$	Tuple size in bytes of table $X$ or attribute $X$



index is built. The storage space of a bitmap index built on attribute  $A$  from table  $T$  is equal to  $\frac{|A||T|}{8}$  bytes [19,20]. Bitmap join indexes are built on dimension table attributes. Each bitmap contains as many bits as the number of tuples in fact table  $F$ . The size of their storage space is then  $S = \frac{|A||F|}{8}$  bytes.

### 3.2 Bitmap Join Index Maintenance Cost

Data updates (mainly insert operations in decisions-support systems) systematically trigger index updates. These operations are applied either on a fact table or dimensions. The cost of updating bitmap join indexes is presented in the following sections.

**Insertion cost in fact table.** Assume a bitmap join index built on attribute  $A$  from dimension table  $T$ . While inserting tuples in fact table  $F$ , it is first necessary to search for the tuple of  $T$  that is able to be joined with them. At worst, the whole table  $T$  is scanned ( $P_T$  pages are read). It is then necessary to update all bitmaps. At worst, all bitmaps are scanned:  $\frac{|A||F|}{8S_p}$  pages are read, where  $S_p$  denotes the size of one disk page. The index maintenance cost is then  $C_{maintenance} = p_T + \frac{|A||F|}{8S_p}$ .

**Insertion cost in dimension tables.** An insertion in dimension  $T$  may induce or not a domain expansion for attribute  $A$ . When not expanding the domain, the fact table is scanned to search for tuples that are able to be joined with the new tuple inserted in  $T$ . This operation requires to read  $p_F$  pages. It is then necessary to update the bitmap index. This requires  $\frac{|A||F|}{8S_p}$  I/Os. When expanding the domain, it is necessary to add the cost of building a new bitmap ( $\frac{|F|}{8S_p}$  pages). The maintenance cost of bitmap join indexes is then  $C_{maintenance} = p_F + (1 + \xi) \frac{|A||F|}{8S_p}$ , where  $\xi$  is equal to one if there is expansion and zero otherwise.

### 3.3 Data Access Cost

We propose two cost models to estimate the number of I/Os needed for data access. In the first model, we do not take any hypothesis about how indexes are physically implemented. In the second model, we assume that access to the index bitmaps is achieved through a b-tree such as is the case in Oracle. Due to lack of space and our experiments under Oracle we only detail here the second model because of running our experiments. however, the first model is not detailed here due to the lack of space.

In this model, we assume that the access to bitmaps is realized through a b-tree (meta-indexing) in which leaf nodes point to bitmaps. The cost, in number of I/Os, of exploiting a bitmap join index for a given query may be written as follows:  $C = C_{descent} + C_{scan} + C_{read}$ , where  $C_{descent}$  denotes the cost needed to reach the leaf nodes from the b-tree root,  $C_{scan}$  denotes the cost of scanning leaf nodes to retrieve the right search key and the cost of reading the bitmaps associated to this key, and  $C_{read}$  finally gives the cost of reading the indexed table's tuples.

The descent cost in the b-tree depends on its height. The b-tree's height built on attribute  $A$  is  $\log_m |A|$ , where  $m$  is the b-tree's order. This order is equal to  $K + 1$ , where  $K$  represents the number of search keys in each b-tree node.  $K$  is equal to  $\frac{S_p}{w(A) + S_{pointer}}$ , where  $w(A)$  and  $S_{pointer}$  are respectively the size of the indexed attribute  $A$  and the size of a disk page pointer in bytes. Without adding the b-tree leaf node level, the b-tree descent cost is then  $C_{descent} = \log_m |A| - 1$ .

The scanning cost of leaf nodes is  $\frac{|A|}{m-1}$  (at worst, all leaf nodes are read). Data access is achieved through bits set to one in each bitmap. In this case, it is necessary to read each bitmap. The reading cost of  $d$  bitmaps is  $d \frac{|F|}{8S_p}$ . Hence, the scanning cost of the leaf nodes is  $C_{scan} = \frac{|A|}{m-1} + d \frac{|F|}{8S_p}$ .

The reading cost of the indexed table's tuples is computed as follow. For a bitmap index built on attribute  $A$ , the number of read tuples is equal to  $\frac{|F|}{|A|}$  (if data are uniformly distributed). Generally, the total number of read tuples for a query using  $d$  bitmaps is  $N_r = d \frac{|F|}{|A|}$ . Knowing the number of read tuples, the number of I/Os in the reading phase is  $C_{read} = p_F(1 - e^{-\frac{N_r}{p_F}})$  [15], where  $p_F$  denotes the number of pages needed for store the fact table.

In summary, the evaluation cost of a query exploiting a bitmap join index is  $C_{index} = \log_m |A| - 1 + \frac{|A|}{m-1} + d \frac{|F|}{8S_p} + p_F(1 - e^{-\frac{N_r}{p_F}})$ .

### 3.4 Join Cost Without Indexes

If the bitmap join indexes are not useful while evaluating a given query, we assume that all joins are achieved by the hash-join method. The number of I/Os needed for joining table  $R$  with table  $S$  is then  $C_{hash} = 3(p_S + p_R)$  [14].

## 4 Bitmap Join Index Selection Strategy

Our index selection strategy proceeds in several steps. The candidate index set is first built from the frequent itemsets mined from the workload (Section 2). A greedy algorithm then exploits an objective function based on our cost models (Section 3) to prune the least advantageous indexes. The detail of these steps and the construction of the objective function are provided in the following sections.

### 4.1 Candidate Index Set Construction

From the frequent itemsets (Section 2) and the data warehouse schema (foreign keys of the fact table, primary keys of the dimensions, etc.), we build a set of candidate indexes.

The SQL statement for building a bitmap join index is composed of three clauses: **On**, **From** and **Where**. The **On** clause is composed of attributes on which is built the index (non-key attributes in the dimensions), the **From** clause contains all joined tables and the **Where** clause contains the join predicates.

We consider a frequent itemset  $\langle Table.attribute_1, \dots, Table.attribute_n \rangle$  composed of elements such as  $Table.attribute$ . Each itemset is analyzed to determine the different clauses of the corresponding index. We first extract the

elements containing foreign keys of the fact table because they are necessary to define the **From** and **Where** index clauses. Next, we retrieve the itemset elements that contain primary keys of dimensions to form the **From** index clause. The elements containing non-key attributes of dimensions form the **On** index clause. If such elements do not exist, the bitmap join index cannot be built.

## 4.2 Objective Functions

In this section, we describe three objective functions to evaluate the variation of query execution cost, in number of I/Os, induced by adding a new index. The query execution cost is assimilated to computing the cost of hash joins if no bitmap join index is used or to the data access cost through indexes otherwise. The workload execution cost is obtained by adding all execution costs for each query within this workload. The first objective function advantages the indexes providing more profit while executing queries, the second one advantages the indexes providing more benefit and occupying less storage space, and the third one combines the first two in order to select at first all indexes providing more profit and then keep only those occupying less storage space when this resource becomes critical. The first function is useful when storage space is not limited, the second one is useful when storage space is small and the third one is interesting when this storage space is quite large. The detail of computing each function is not given due to the lack of space.

## 4.3 Index Configuration Construction

The index selection algorithm is based on a greedy search within the candidate index set  $I$  given as an input. The objective function  $F$  must be one of the functions: profit ( $P$ ), profit/space ratio ( $R$ ) or hybrid ( $H$ ). If  $R$  is used, we add to the algorithm's input the space storage  $M$  allotted for indexes. If  $H$  is used, we also add threshold  $\alpha$  as input.

In the first algorithm iteration, the values of the objective function are computed for each index within  $I$ . The execution cost of all queries in workload  $Q$  is equal to the total cost of hash joins. The index  $i_{max}$  that maximizes  $F$ , if it exists, is then added to the set of selected indexes  $S$ . If  $R$  or  $H$  is used, the whole space storage  $M$  is decreased by the amount of space occupied by  $i_{max}$ .

The function values of  $F$  are then recomputed for each remaining index in  $I - S$  since they depend on the selected indexes present in  $S$ . This helps taking into account the interactions that probably exist between the indexes. We repeat these iterations until there is no improvement or all indexes have been selected ( $I - S = \emptyset$ ). If functions  $R$  or  $H$  are used, the algorithm also stops when storage space is full.

## 5 Experiments

In order to validate our bitmap join index selection strategy, we have run tests on a data warehouse implemented within Oracle 9i, on a Pentium 2.4 GHz PC

with a 512 MB main memory and a 120 GB IDE disk. This data warehouse is composed of the fact table **Sales** and five dimensions **Customers**, **Products**, **Promotions**, **Times** and **Channels**. We have measured for different value of the minimal support parameterized in Close the workload execution time. In practice, the minimal support limits the number of candidate indexes to generate and selects only those that are frequently used.

For computing the different costs from our models, we fixed the value of  $S_p$  (disk page size) and  $S_{pointer}$  (page pointer size) to 8 MB and 4 MB respectively. These values are those indicated in the Oracle 9i configuration file. The workload is composed of forty decision-support queries containing several joins. We measured the total execution time when building indexes or not. In the case of building indexes, we also measured the total execution time when we applied each objective function among of profit, ratio profit/space and hybrid. We also measured the disk space occupied by the selected indexes. When applying the cost models, we reduce the number of indexes and thereby the storage space needed to store these indexes.

**Profit function experiment.** Figure 2 shows that the selected indexes improve query execution time with and without application of our cost models until the minimal support forming frequent itemsets reaches 47.5%. Moreover, the execution time decreases continuously when the minimal support increases because the number of indexes decreases. For high values of the minimal support (greater than 47.5%), the execution time is closer to the one obtained without indexes. This case is predictable because there is no or few candidate indexes to create. The maximal gain in time in both cases is respectively 30.50% and 31.85%. Despite of this light drop of 1.35% in time gain when the cost models are used (fewer indexes are built), we observe a significant gain in storage space (equal to 32.79% in the most favorable case) as shown in figure 3. This drop in number of indexes is interesting when the data warehouse update frequency is high because update time is proportional to the number of indexes. On the

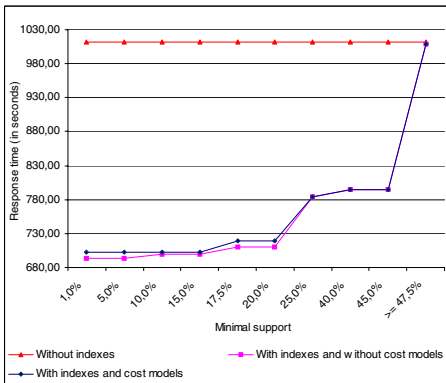


Fig. 2. Profit function

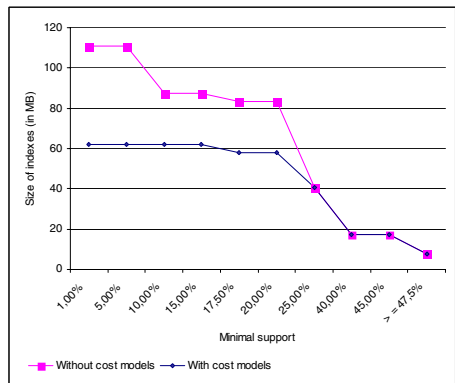


Fig. 3. Index storage space

other hand, the gain in storage space helps limiting the storage space allotted for indexes by the administrator.

**Profit/space ratio function experiment.** In these experiments, we have fixed the value of minimal support to 1%. This value gives the highest number of frequent itemsets and consequently the highest number of candidate indexes. This helps varying storage space within a wider interval. We have measured query execution time according to the percentage of storage space allotted for indexes. This percentage is computed from the space occupied by all indexes. Figure 4 shows that execution time decreases when storage space occupation increases. This is predictable because we create more indexes and thus better improve the execution time. We also observe that the maximal time gain is equal to 28.95% and it is reached for a space occupation of 59.64%. This indicates that if we fix space storage to this value, we obtain a time gain close to the one obtained with the profit objective function (30.50%). This case is interesting when the administrator does not have enough space to store all the indexes.

**Hybrid function experiment.** We repeated the previous experiments with the hybrid objective function. We varied the value of parameter  $\alpha$  between 0.1 and 1 by 0.1 steps. The obtained results with  $\alpha \in [0.1, 0.7]$  and  $\alpha \in [0.8, 1]$  are respectively equal to those obtained with  $\alpha = 0.1$  and  $\alpha = 0.7$ . Thus, we represent in figure 5 only the results obtained with  $\alpha = 0.1$  and  $\alpha = 0.7$ . This figure shows that for  $\alpha = 0.1$ , the results are close to those obtained with profit/space ratio the function ; and for  $\alpha = 0.8$ , they are close to those obtained with the profit function. The maximal gain in execution time is respectively equal to 28.95% and 29.95% for  $\alpha = 0.1$  and  $\alpha = 0.8$ . We explain these results by the fact that bitmap join indexes built on several attributes need more storage space. However, as they pre-compute more joins, they better improve the execution time. The space storage allotted for indexes then fills up very quickly after a few iterations of the greedy algorithm. This explains why the parameter  $\alpha$  does not significantly affect our algorithm and the experiment results.



Fig. 4. Profit/space ratio function

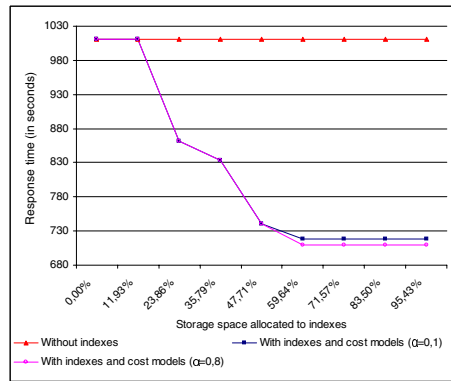


Fig. 5. Hybrid function

## 6 Conclusion and Perspectives

In this article, we presented an automatic strategy for bitmap index selection in data warehouses. This strategy first exploits frequent itemsets obtained by the Close algorithm from a given workload to build a set of candidate bitmap join indexes. With the help of cost models, we keep only the most advantageous candidate indexes. These models estimate data access cost through indexes, as well as maintenance and storage cost for these indexes. We have also proposed three objective functions: profit, profit/space ratio and hybrid that exploit our cost models to evaluate the execution cost of all queries. These functions are themselves exploited by a greedy algorithm that recommends a pertinent configuration of indexes. This helps our strategy respecting constraints imposed by the system (limited number of indexes per table) or the administrator (storage space allotted for indexes). Our experimental results show that the application of cost models to our index selection strategy decreases the number of selected indexes without a significant loss in performance. This decrease actually guarantees a substantial gain in storage space, and thus a decrease in maintenance cost during data warehouse updates.

Our work shows that the idea of using data mining techniques for data warehouse auto-administration is a promising approach. It opens several future research axes. First, it is essential to keep on experimenting in order to better evaluate system overhead in terms of index building and maintenance. It could also be very interesting to compare our approach to other index selection methods. Second, extending our approach to other performance optimization techniques (materialized views, buffering, physical clustering, etc.) is another promising perspective. Indeed, in a data warehouse environment, it is principally in conjunction with other physical structures such as materialized views that indexing techniques provide significant gains in performance. For example, our context extraction may be useful to build clusters of queries that maximize the similarity between queries within each cluster. Each cluster may be then a starting point to materialize views. In addition, it could be interesting to design methods to efficiently share the available storage space between indexes and views.

## References

1. S. Agrawal, S. Chaudhuri, and V. Narasayya. Automated selection of materialized views and indexes in SQL databases. In *26th International Conference on Very Large Data Bases (VLDB 2000)*, Cairo, Egypt, pages 496–505, 2000.
2. S. Agrawal, S. Chaudhuri, and V. Narasayya. Materialized view and index selection tool for Microsoft SQL Server 2000. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, Santa Barbara, USA, page 608, 2001.
3. K. Aouiche, J. Darmont, and L. Gruenwald. Frequent itemsets mining for database auto-administration. In *7th International Database Engineering and Application Symposium (IDEAS 2003)*, Hong Kong, China, pages 98–103, 2003.
4. S. Chaudhuri, M. Datar, and V. Narasayya. Index selection for databases: A hardness study and a principled heuristic solution. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1313–1323, 2004.

5. Y. Feldman and J. Reouven. A knowledge-based approach for index selection in relational databases. *Expert System with Applications*, 25(1):15–37, 2003.
6. S. Finkelstein, M. Schkolnick, and P. Tiberio. Physical database design for relational databases. *ACM Transactions on Database Systems*, 13(1):91–128, 1988.
7. M. Frank, E. Omiecinski, and S. Navathe. Adaptive and automated index selection in RDBMS. In *3rd International Conference on Extending Database Technology (EDBT 1992)*, Vienna, Austria, volume 580 of *Lecture Notes in Computer Science*, pages 277–292, 1992.
8. M. Golfarelli, S. Rizzi, and E. Saltarelli. Index selection for data warehousing. In *4th International Workshop on Design and Management of Data Warehouses (DMDW 2002)*, Toronto, Canada, pages 33–42, 2002.
9. H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index selection for OLAP. In *13th International Conference on Data Engineering (ICDE 1997)*, Birmingham, U.K., pages 208–219, 1997.
10. W. Inmon. *Building the Data Warehouse*. John Wiley & Sons, third edition, 2002.
11. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, second edition, 2002.
12. J. Kratica, I. Ljubić, and D. Tošić. A genetic algorithm for the index selection problem. In *Applications of Evolutionary Computing, Essex, England*, volume 2611 of *LNCS*, pages 281–291, 2003.
13. W. Labio, D. Quass, and B. Adelberg. Physical database design for data warehouses. In *13th International Conference on Data Engineering (ICDE 1997)*, Birmingham, U.K., pages 277–288, 1997.
14. P. Mishra and M. Eich. Join processing in relational databases. *ACM Computing Surveys*, 24(1):63–113, 1992.
15. P. O’Neil and D. Quass. Improved query performance with variant indexes. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 1997)*, Tucson, USA, pages 38–49, 1997.
16. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *7th International Conference on Database Theory (ICDT 1999)*, Jerusalem, Israel, volume 1540 of *LNCS*, pages 398–416, 1999.
17. S. Sarawagi. Indexing OLAP data. *Data Engineering Bulletin*, 20(1):36–43, 1997.
18. G. Valentin, M. Zuliani, D. Zilio, G. Lohman, and A. Skelley. DB2 advisor: An optimizer smart enough to recommend its own indexes. In *16th International Conference on Data Engineering (ICDE 2000)*, San Diego, USA, pages 101–110, 2000.
19. M. Wu. Query optimization for selections using bitmaps. In *ACM SIGMOD International Conference on Management of Data (SIGMOD 1999)*, Philadelphia, USA, pages 227–238, 1999.
20. M. Wu and A. Buchmann. Encoded bitmap indexing for data warehouses. In *14th International Conference on Data Engineering (ICDE 1998)*, Orlando, USA, pages 220–230, 1998.

# A Survey of Open Source Tools for Business Intelligence

Christian Thomsen and Torben Bach Pedersen

Department of Computer Science, Aalborg University

**Abstract.** The industrial use of open source Business Intelligence (BI) tools is not yet common. It is therefore of interest to explore which possibilities are available for open source BI and compare the tools.

In this survey paper, we consider the capabilities of a number of open source tools for BI. In the paper, we consider three Extract-Transform-Load (ETL) tools, three On-Line Analytical Processing (OLAP) servers, two OLAP clients, and four database management systems (DBMSs). Further, we describe the licenses that the products are released under.

It is argued that the ETL tools are still not very mature for use in industry while the DBMSs are mature and applicable to real-world projects. The OLAP servers and clients are not as powerful as commercial solutions but may be useful in less demanding projects.

## 1 Introduction

The use of Business Intelligence tools is popular in industry [25,29,30]. However, the use of open source tools seems to be limited. The dominating tools are closed source and commercial (see for example [25] for different vendors' market shares for OLAP servers). Only for database management systems (DBMSs), there seems to be a market where open source products are used in industry, including business-critical systems such as online travel booking, management of subscriber inventories for tele communications, etc. [23]. Thus, the situation is quite different from, for example, the web server market where open source tools as Linux and Apache are very popular [38].

To understand the limited use of open source BI tools better, it is of interest to consider which tools are available and what they are capable of. This is the purpose of this paper. In the European Internet Accessibility Observatory (EIAO) project, where accessibility data is collected, it is intended to build a BI solution based on open source software. It is therefore of relevance for this project to investigate the available products.

In the survey we will consider products for making a complete solution with an Extract-Transform-Load (ETL) tool that loads data into a database managed by a DBMS. On top of the DBMS, an On-Line Analytical Processing (OLAP) server providing for fast aggregate queries will be running. The user will be communicating with the OLAP server by means of an OLAP client. We limit ourselves to these kinds of tools and do not consider, for example, data mining tools or Enterprise Application Integration (EAI) tools. Use of data mining tools would also be of relevance in many BI settings, but data mining is a more



advanced feature which should be considered in future work. EAI tools may have some similarities with ETL tools, but are more often used in online transactional processing (OLTP) systems.

The rest of the paper is structured as follows. Section 2 gives a primer on open source licenses. Section 3 presents the criteria used in the evaluation of the different tools. Section 4 considers ETL tools. Section 5 deals with OLAP servers, while Section 6 deals with OLAP clients. Finally, we consider DBMSs in Section 7 before concluding and pointing to future work in Section 8.

## 2 Open Source Licenses

To make the findings on licenses more comprehensible, we include a description of the open source licenses that will be referred to later in the paper. The *GNU General Public License (GPL)* [11] is a classic, often used open source license. Any user is free to make changes to the source code. If the changed version is only used privately, it is not a requirement that its source code is released. If it, however, is distributed in some way, then the source code must be made available under the GPL (i.e. also released as open source that any user is free to change and copy). It should be noted that a library released under the GPL will require any program that uses it to be licensed under the GPL. This is not the case when using *GNU Library General Public License (LGPL)* [12] which apart from that is much like the GPL. The *Common Public License (CPL)* [7] was developed by IBM as an open source license. Like the GPL, the CPL requires that the source code for a modified version of a program is made publicly available if the new version is distributed to anyone. Programs and libraries released under the CPL may be used from and integrated with software released under other (also closed source) licenses.

The *Mozilla Public License* [22] is also an open source license that requires the code for any distributed modified works to be made publicly available. It is allowed to use a library under the Mozilla Public License from a closed source application. Thus the license has some similarities with the LGPL. The *Apache License* [1], [2] allows the code to be used both in open source, free programs and in commercial programs. It is also possible to modify the code and redistribute it under another license under certain conditions (e.g. the use of the original code should be acknowledged). Version 1.0 and 1.1 of the Apache License [1] included requirements about the use of the name “Apache” in documentation and advertising materials. That meant that the license should be modified for use in non-Apache projects. This was changed with version 2.0 [2]. The *BSD License* [5] is a very liberal open source license. It is permitted to use source code from a BSD licensed program in a commercial, closed source application. As long as any copyright notices remain in the modified code, there are no requirements saying that modifications of the code should be BSD licensed or open source.

## 3 Conduct of the Survey

In this section, we present the criteria used for the evaluation of the considered products. The criteria with a technical nature have been inspired by the

functionality offered by the leading commercial BI tools. Other criteria, such as the type of license, are interesting when looking at open source tools. Based on the criteria given below, we collected data about the products (all found on the Internet) by examining their source code, available manuals, and homepages including any forums. The findings were collected in Nov-Dec. 2004.

**Criteria for All Categories.** When deciding between different products, a potential user would most often prefer a product that is compatible with his<sup>1</sup> existing operating system and hardware. Thus, for all the products, it is of interest to investigate which hardware and software platforms the tools are available for. In this survey we will only look at open source products. As described in Section 2 there are, however, many different open source licenses that have different permissions and restrictions. Therefore, the license used by a product is also of great interest.

**Criteria for ETL Tools.** When comparing ETL tools, there are several criteria to consider. First, it should be considered which data sources and targets a given tool supports. Here, it should be considered whether the tool is for loading data into ROLAP or MOLAP systems, i.e. into relational tables or multidimensional cubes [30]. In many practical applications, it should be possible to extract data from different sources and combine the data in different ways. Further, it should be possible to load the data into different tables/cubes. Therefore support for these issues should be considered. It should also be considered which types of data sources an ETL tool can extract data from and whether it supports incremental load in an automatic fashion (not requiring two separate flows to be specified). It is also of interest how the ETL process is specified by the user, i.e. whether a graphical user interface (GUI) exists and if the user can specify the process directly by means of some specification language. Another important issue for ETL tools is their capabilities for data cleansing. Here it should be considered how data cleansing is supported, i.e. if predefined methods exist and how the user can specify his own rules for data cleansing.

**Criteria for OLAP Servers.** For an OLAP server it is of interest to know how it handles data. It should thus be considered whether the tool is ROLAP, MOLAP, or HOLAP oriented, where HOLAP is short for Hybrid OLAP [30]. Further, it is of interest if the product is capable of handling large data sets (for example, data sets greater than 10 gigabytes). It should also be taken into account whether an OLAP server has to be used with a specific DBMS or if it is independent of the underlying DBMS. Precomputed aggregates can in many situations lead to significant performance gains. It is therefore relevant to see whether an OLAP server can use aggregates and if so, whether the user can specify which aggregates to use. Finally, it is of relevance to investigate which application programming interfaces (APIs) and query languages an OLAP server supports. A product that uses standards or de-facto standards is much more useful with other tools than a product using a non-standard API or query language.

---

<sup>1</sup> We use “his” as short for “his/her”.

**Criteria for OLAP Clients.** For an OLAP client it should be considered which OLAP server(s) the OLAP client can be used with. As for OLAP servers, it should also be taken into account which API(s) and query language(s) the OLAP client supports. With respect to reports, it is interesting to see if the OLAP client supports prescheduled reports, perhaps through a server component. If so, the user could, for example, make the OLAP client generate a sales report every Friday afternoon. When a report has been generated (manually or as prescheduled report), it is often useful to be able to export the report to some common format that could be emailed to someone else. Therefore, it should be investigated which export facilities an OLAP client offers. In generated reports, different types of graphs are often used. It should thus also be considered how well an OLAP clients supports different kinds of graphs.

**Criteria for DBMSs.** There are many possible criteria to consider for DBMSs. In this survey we will, however, only look at criteria directly relevant for BI purposes. First of all, a DBMS should be capable of handling large data sets if the DBMS is to be used in BI applications. Thus this is an issue to investigate. When choosing a DBMS for BI, it is also of relevance which performance improving features the DBMS offers. In this survey we will look into the support for materialized views that can yield significant performance gains for precomputed aggregates. Many commercial ROLAP systems use bitmap indices to achieve good performance [30]. It is also of interest to find out whether these are supported in the considered products. Further, in a typical schema for a data warehouse, star joins may be a faster to use and, thus, the support for these is an issue. Finally, we will consider partitioning which can yield performance improvements and replication which may improve performance and reliability.

## 4 ETL Tools

In this section, we will consider the three ETL tools Bee, CloverETL, and Octopus. These were all the available tools we found. We found many other open source ETL projects that were not carrying any implementation, but more or less only stated objectives. Examples of such projects are OpenSrcETL [27] and OpenETL [26]. Another disregarded project, was cplusql [8] which had some source code available, but for which we did not find any other information.

**Bee.** Bee version 1.1.0 [3] is a package consisting of an ETL tool, an OLAP server, and an OLAP client web interface. The ETL tool and the OLAP server of Bee are ROLAP oriented. Bee is available under both an open source GPL license and a commercial license. Bee is implemented mainly in Perl with parts implemented in C. Therefore, the access to data is provided by the Perl module DBI. Bee comes with its own driver for comma-separated files. To extract data, Bee needs a small server application (included) to be running on the host holding the data. Bee is primarily written for Linux, but is also running on Windows platforms. The mentioned server application needed for extracting data runs on different varieties of UNIX and Windows. The ETL process can be specified by

means of a GUI. The GUI will create an XML file defining the process. It is thus also possible for the user to use Bee without using the GUI by creating the XML file manually. It is possible to have several flows and combine them. It is also possible to insert into more than one table in the database. There seems to be no support for automatic incremental loading. The possibility for data cleansing is introduced by means of allowing the user to write custom transformations in Perl. A standard library of transformations is not included. Thus the user needs to program any needed transformation.

**CloverETL.** CloverETL version 1.1.2 [6] is also a ROLAP oriented ETL tool. Parts of it are distributed under the GPL license whereas other parts are distributed under the LGPL license. CloverETL is implemented in Java and uses JDBC to transfer data. The ETL process is specified in an XML file. In this XML file, a directed graph representing the flow must be described. Currently, CloverETL does not include a GUI, but work is in progress with respect to this. CloverETL supports combination of several flows as well as import to several tables in the database. There is no support for automatic incremental load. With respect to cleansing, CloverETL supports insertion of a default value, but apart from this, the user will have to implement his own transformations in Java.

**Octopus.** Octopus version 3.0.1 [9] is a ROLAP oriented ETL tool under the LGPL license. It is implemented in Java and is capable of transferring data between JDBC sources. Octopus is bundled with JDBC drivers for XML and comma-separated files. Further, it is possible to make Octopus create SQL files with insert and DDL statements that can be used for creating a database holding the considered data. Like Bee, Octopus is shipped with a GUI that creates an XML file specifying the ETL process. Octopus can also be used without the GUI and as a library. Octopus is created for transferring data between one JDBC source and another. It is apparently not possible to combine data from one database with data extracted from another. It is possible to extract data from more than one table in the same database as well as insert into more than one table in the target database. There is no direct support for automatic incremental loading. Basic data cleansing functionality is provided. It is possible to make Octopus insert a default value, shorten too long strings, replace invalid foreign key values, find and replace values, do numeric conversions, and change date formats. These cleansings are done by predefined transformations. The user can also implement transformations on his own in Java and JavaScript.

**General Comments.** The considered open source ETL tools are still not as powerful as one could wish. For example, most of the data cleansing to be done must be coded by the user (with the exception of Octopus which provides some default transformations for very basic data cleansing). Further, the products do not support automatic incremental load which would be very useful for everyday use of the products. In general, the quality of the documentation for the described products is not very good or comprehensive. Further, not much documentation is available. An exception is again Octopus for which a manual of more than 120 pages is available. However, also this manual is not complete. For example, it

explains how to set up which *logger* to use but does not tell about the differences between the available loggers. Thus the quality of the open source ETL products is still not as high as the quality of many commercially available products. Indeed it would probably be difficult to use one of the open source ETL tools for a demanding load job in an enterprise data warehouse environment.

## 5 OLAP Servers

In this section, we will consider the three OLAP servers Bee, Lemur, and Mondrian. Another possible candidate for consideration would be pocOLAP [31] which, however, in the documentation is said not to be an OLAP server. It provides access to data from DBMS through a web-interface but is not intended provide advanced OLAP functionality or real-time data analysis. OpenRolap [28] is a related tool which generates aggregate tables for a given database. Apart from these tools we did not find any candidates. As for the ETL tool category, there exist other projects that currently carry no code, but only state objectives. An example of such a project is gnuOLAP [13].

**Bee.** The OLAP server of the Bee package is, as previously stated, a ROLAP oriented server. It uses a MySQL system to manage the underlying database and aims to be able to handle up to 50GB of data efficiently [4]. Despite this, it does not seem to be possible to choose which precomputed aggregates to use. From the documentation, it is not clear which query language(s) and API(s) Bee supports. In general, there is not much English documentation available for Bee, neither from the homepage [3], nor in the downloadables.

**Lemur.** Unlike the other OLAP servers considered in this paper, Lemur [17] is a HOLAP oriented OLAP server. It is released under the GPL license and is written in C++ for Linux platforms, but is portable. Lemur is a product under development and still has no version number. The homepage for the Lemur project [17] states that for now, the primary goal is to support the developers research interests and that their goals are believed to be too ambitious to deliver usable code now. This is also reflected in the fact that the API is still being designed and in reality is not available for use from outside the Lemur package. Further the user would need to implement methods to load data from a database on his own. It is also not possible to specify the aggregates to be used. No information on how well Lemur scales when applied to large data sets has been found. In summary, the Lemur project is not of much practical use for industry projects so far. However, the goal of eventually producing a HOLAP oriented server outperforming Mondrian (see below) is interesting.

**Mondrian.** Mondrian 1.0.1 [19] is an OLAP server implemented in Java. It is ROLAP oriented and can, unlike Bee, be used with any DBMS for which a JDBC driver exists. Mondrian is released under the CPL license [7]. The current version of Mondrian has an API that is similar to ADO MD from Microsoft [20]. Support for the standard APIs JOLAP [14] and XMLA [39] is planned. Further, the MDX query language [35], known from Microsoft's products, is supported by

Mondrian. By default Mondrian will use some main memory for caching results of aggregation queries. It is, however, neither possible for the user to specify what should be cached nor which aggregates should exist in the database. The documentation states that Mondrian will be able to handle large data sets if the underlying DBMS is, since all aggregation is done by the DBMS.

**General Comments.** The Mondrian OLAP server seems to be the best of the described products. Lemur is for the time being not usable for real world applications while it is difficult to judge Bee because of its lack of English documentation. Mondrian is, however, a usable product which works with JDBC-enabled DBMSs. For none of the products, it seems possible to choose which aggregates to use. In most environments this feature would result in significant performance improvements.

## 6 OLAP Clients

In this section, we will describe the OLAP clients Bee and JPivot. These were the found open source OLAP clients that are actually implemented.

**Bee.** The Bee project also provides an OLAP client. The client is web-based and is used with the Bee OLAP server. Currently, Microsoft Internet Explorer and Mozilla browsers are explicitly supported in the downloadable code. Again, it has not been possible to determine which API(s) and query language(s) Bee supports. The Bee OLAP client can interactively present multidimensional data by means of Virtual Reality Modeling Language (VRML) technology [37]. Bee can generate different types of graphs (pie, bar, chart, etc.) in both 2D and 3D. It is possible to export data from Bee to Excel, Portable Document Format (PDF), Portable Networks Graphics (PNG), PowerPoint, text, and Extensible Markup Language (XML) formats. Connection with the statistical package R [33] is also evaluated. It does not seem to be possible to preschedule reports.

**JPivot.** JPivot version 1.2.0 [16] is a web-based OLAP client for use with the Mondrian OLAP server. However, the architecture should allow for later development of a layer for XMLA [39]. As Mondrian, JPivot uses MDX as its query language. It is written in Java and JSP. JPivot generates graphs by means of JFreeChart [15] which provides different kinds of 2D and 3D graphs. With respect to export of reports, JPivot is limited to Portable Document Format (PDF) and Excel format. Support for prescheduled reports has not been found. JPivot is released under a license much like the Apache Software License Version 1.1 [1] (but without restrictions regarding the use of the name “Apache”). However, other software packages are distributed with JPivot and have other software licenses, e.g. JFreeChart which uses the LGPL license.

**General Comments.** Both the considered OLAP clients are to be used with specific OLAP servers, namely Bee with Bee and JPivot with Mondrian. Both of them are web-based such that specific software does not have to be installed at client machines already equipped with a browser. Both products are capable of

exporting generated reports to other commonly used file formats such as PDF, but neither of them supports prescheduled reports.

## 7 DBMSs

In this section we consider four open source DBMSs: MonetDB, MySQL, MaxDB, and PostgreSQL. Other open source DBMSs are available, but these four were chosen as they are the most visible, well-known high-performance DBMSs.

**MonetDB.** MonetDB, currently in version 4.4.2, is developed as a research project at CWI. MonetDB is “designed to provide high performance on complex queries against large databases, e.g. combining tables with hundreds of columns and multi-million rows” [21]. To be efficient, MonetDB is, among other techniques, exploiting CPU caches and full vertical fragmentation (however, the fragments must be placed on the same disk). It thus uses very modern and often hardware-near approaches to be fast. MonetDB is mainly implemented in C with some parts in C++. It is available for 32- and 64-bit versions of Linux, Windows, MacOS X, Sun Solaris, IBM AIX, and SGI IRIX. MonetDB comes with a license like the Mozilla Public License (references to “Mozilla” are replaced by references to “MonetDB”) [21]. With respect to features often usable in a BI context, it is interesting to notice that MonetDB does not support bitmap indices, materialized views (normal views are supported), replication, or star joins. However, this does not mean that MonetDB is not usable for BI purposes. On the contrary, MonetDB has been successfully applied in different BI contexts [21]. Currently, the developers are working on improving the scalability for OLAP and data mining in the 64-bit versions of MonetDB.

**MySQL.** MySQL is a very popular open source database with more than five millions installations [24]. The latest production release is version 4.1, and version 5.0 is in the alpha stage. MySQL is implemented in C and C++ and is available for a large variety of 32- and 64-bits platforms. Users of MySQL can choose between an open source GPL license and a commercial license that gives permissions not given by the GPL license. For BI purposes, MySQL lacks support of materialized views (even ordinary views are not available until version 5.0), bitmap indices and star joins. However, one-way replication (i.e. one master, several slaves) is supported and partitioning is to some degree supported by the *NDB Cluster* (NDB is a name, not an acronym) on some of the supported platforms. Further, MySQL is capable of handling data sets with terabytes of data as documented in case studies available from [24].

**MaxDB.** MaxDB [18] version 7.5 is another RDBMS distributed by the company MySQL AB which also develops MySQL. Formerly, MaxDB was known as SAP DB (developed by SAP AG). MaxDB is developed to be used for OLTP and OLAP in demanding environments with thousands of simultaneous users. It is implemented in C and C++ and is available for most major hardware plat-

forms and operating system environments. As MySQL, it is licensed under two licenses such that users can choose between an open source license (GPL) or a commercial. MaxDB is designed to scale to databases in the terabyte sizes, but there is no user controlled partitioning. It is, however, possible to specify several physical locations for storage of data, and MaxDB will then automatically divide table data between these partitions. There is no support for materialized views (ordinary views are supported), bitmap indexes, or star joins. MaxDB supports one-way replication, also with MySQL such that either of them can be the master.

**PostgreSQL.** PostgreSQL [32] is also a very popular open source DBMS. At the time of this writing, version 8.0 is just about to be released. PostgreSQL is implemented in C and has traditionally only been available for UNIX platforms. From version 8.0, Windows is, however, natively supported. Originally, PostgreSQL is based on the POSTGRES system [36] from Berkeley and has kept using a BSD license. PostgreSQL supports large data sets (installations larger than 32 terabytes exist) and one-way replication. A multiway solution for replication is planned. There is no support for partitioning, bitmap indices or materialized views (ordinary non-materialized views are supported). However, materializations of views may be done in PostgreSQL by using handcoded triggers and procedures [10]. Further, in a research project at North Carolina State University, materialized views are integrated into a derived version of PostgreSQL [34]. Bitmap indices are planned to be supported in a future release.

**General Comments.** The considered DBMSs have different strengths and weaknesses, and so there is not a single of them to be chosen as *the best*. In general, these open source products support more advanced features such as partitioning and replication. Further, the DBMSs are capable of handling very large data sets, are available for a number of platforms, and are very reliable. The category of DBMSs is thus the most *mature* of the considered categories.

## 8 Conclusion and Future Work

Of the considered categories of open source tools (ETL tools, OLAP clients, OLAP servers, and DBMSs), DBMSs are the most mature. They offer advanced features and are applicable to real-world situations where large data sets must be handled with good performance. The ETL tools are the least mature. They do still not offer nearly the same functionality as proprietary products. With respect to the OLAP servers, there is a great difference in their maturity. A product like Lemur is still very immature, while a product like Mondrian is usable in real-world settings. However, important features, such as the opportunity to specify which aggregates to use, are still missing. The OLAP clients are also usable in practical applications. However, they are not very general and can only be used with specific OLAP servers. In general, one of the largest problems for many of the tools is the lack of proper documentation, often making it very difficult to decide how a specific task is performed in a given product.



If one were to create a complete BI installation with open source tools, it would probably be created with JPivot and Mondrian as OLAP client and server, respectively. Which one of the DBMSs should be used would depend on the situation. The ETL tool would then probably be CloverETL, if one did not handcode a specialized tool for the installation. In many BI installations, data mining solutions would also be interesting to apply. The available open source data mining applications should therefore be explored in future work.

## Acknowledgements

This work was supported by the European Internet Accessibility Observatory (EIAO) project, funded by the European Commission under Contract no. 004526.

## References

1. Apache SW License v1.1, [www.apache.org/licenses/LICENSE-1.1](http://www.apache.org/licenses/LICENSE-1.1) as of 05/30/05.
2. Apache SW License, v2.0, [www.apache.org/licenses/LICENSE-2.0](http://www.apache.org/licenses/LICENSE-2.0) as of 05/30/05.
3. Bee, [sourceforge.net/projects/bee/](http://sourceforge.net/projects/bee/) as of 05/30/05.
4. Bee Project, [bee.insightstrategy.cz/en/](http://bee.insightstrategy.cz/en/) as of 05/30/05.
5. BSD license, [opensource.org/licenses/bsd-license.php](http://opensource.org/licenses/bsd-license.php) as of 05/30/05.
6. CloverETL, [cloveretl.berlios.de](http://cloveretl.berlios.de) as of 05/30/05.
7. CPL v1.0, [opensource.org/licenses/cpl1.0.php](http://opensource.org/licenses/cpl1.0.php) as of 05/30/05.
8. cplusql ETL tool, [sourceforge.net/projects/cplusql/](http://sourceforge.net/projects/cplusql/) as of 05/30/05.
9. Enhydra Octopus, [octopus.objectweb.org/](http://octopus.objectweb.org/) as of 05/30/05.
10. J. Gardner. Materialized Views in PostgreSQL, [jonathangardner.net/PostgreSQL/materialized\\_views/matviews.html](http://jonathangardner.net/PostgreSQL/materialized_views/matviews.html) as of 05/30/05.
11. GNU GPL License, [www.gnu.org/copyleft/gpl.html](http://www.gnu.org/copyleft/gpl.html) as of 05/30/05.
12. GNU LGPL License, [www.gnu.org/copyleft/lgpl.html](http://www.gnu.org/copyleft/lgpl.html) as of 05/30/05.
13. gnuOLAP, [gnuolap.sourceforge.net/](http://gnuolap.sourceforge.net/) as of 05/30/05.
14. Java OLAP Interface (JOLAP), [www.jcp.org/en/jsr/detail?id=69](http://www.jcp.org/en/jsr/detail?id=69) as of 05/30/05.
15. JFreeChart, [www.jfree.org/jfreechart](http://www.jfree.org/jfreechart) as of 05/30/05.
16. Jpivot, [jpivot.sourceforge.net/](http://jpivot.sourceforge.net/) as of 05/30/05.
17. Lemur OLAP library, [www.nongnu.org/lemur/](http://www.nongnu.org/lemur/) as of 05/30/05.
18. MaxDB, [www.mysql.com/products/maxdb/](http://www.mysql.com/products/maxdb/) as of 05/30/05.
19. Mondrian components, [perforce.eigenbase.org:8080/open/mondrian/doc/index.html](http://perforce.eigenbase.org:8080/open/mondrian/doc/index.html), as of 05/30/05.
20. Mondrian FAQs, [perforce.eigenbase.org:8080/open/mondrian/doc/faq.html](http://perforce.eigenbase.org:8080/open/mondrian/doc/faq.html), as of 05/30/05.
21. MonetDB, [monetdb.cwi.nl/](http://monetdb.cwi.nl/) as of 05/30/05.
22. Mozilla Public License v1.1, [www.mozilla.org/MPL/MPL-1.1.html](http://www.mozilla.org/MPL/MPL-1.1.html) as of 05/30/05.
23. MySQL Case Studies, [www.mysql.com/it-resources/case-studies/](http://www.mysql.com/it-resources/case-studies/) as of 05/30/05.
24. MySQL Database Server, [www.mysql.com/products/mysql/](http://www.mysql.com/products/mysql/) as of 05/30/05.
25. The OLAP Report Market share analysis, [www.olapreport.com/market.htm](http://www.olapreport.com/market.htm) as of 05/30/05.
26. OpenETL, [openetl.tigris.org](http://openetl.tigris.org) as of 05/30/05.
27. Open Source ETL (OpnSrcETL), [opnsrctel.sourceforge.net/](http://opnsrctel.sourceforge.net/) as of 05/30/05.

28. OpenROLAP, [openrolap.sourceforge.net](http://openrolap.sourceforge.net) as of 05/30/05.
29. T. B. Pedersen. How Is BI Used in Industry?: Report from a Knowledge Exchange Network. In *DaWaK*, pp. 179–188, 2004.
30. T. B. Pedersen and C. S. Jensen. Multidimensional Database Technology. *IEEE Computer*, 34(12):40–46, 2001.
31. pocOLAP - the "little" OLAP-project, [pocolap.sourceforge.net/](http://pocolap.sourceforge.net/) as of 05/30/05.
32. PostgreSQL, [www.postgresql.org](http://www.postgresql.org) as of 05/30/05.
33. The R Project for Statistical Computing, [www.r-project.org](http://www.r-project.org) as of 05/30/05.
34. Self-Organizing Databases, [research.csc.ncsu.edu/selftune/](http://research.csc.ncsu.edu/selftune/) as of 05/30/05.
35. G. Spofford and E. Thomsen. *MDX Solutions: With Microsoft SQL Server Analysis Services*, Wiley, 2001.
36. M. Stonebraker and L. Rowe. The Postgres papers, TR, UC Berkeley, 1987.
37. J. Vacca. *VRML: Bringing Virtual Reality to the Internet*. Academic Press, 1997.
38. Web Server Survey Achives, [news.netcraft.com/archives/web\\_server\\_survey.html](http://news.netcraft.com/archives/web_server_survey.html) as of 05/30/05.
39. XML for analysis, [xmlla.org/download.asp?id=2](http://xmlla.org/download.asp?id=2) as of 05/30/05.

# DWEB: A Data Warehouse Engineering Benchmark

Jérôme Darmont, Fadila Bentayeb, and Omar Boussaïd

ERIC, University of Lyon 2,  
5 av. Pierre Mendès-France,  
69676 Bron Cedex, France

{jdarmont, boussaïd, bentayeb}@eric.univ-lyon2.fr

**Abstract.** Data warehouse architectural choices and optimization techniques are critical to decision support query performance. To facilitate these choices, the performance of the designed data warehouse must be assessed. This is usually done with the help of benchmarks, which can either help system users comparing the performances of different systems, or help system engineers testing the effect of various design choices. While the TPC standard decision support benchmarks address the first point, they are not tuneable enough to address the second one and fail to model different data warehouse schemas. By contrast, our Data Warehouse Engineering Benchmark (DWEB) allows to generate various ad-hoc synthetic data warehouses and workloads. DWEB is fully parameterized to fulfill data warehouse design needs. However, two levels of parameterization keep it relatively easy to tune. Finally, DWEB is implemented as a Java free software that can be interfaced with most existing relational database management systems. A sample usage of DWEB is also provided in this paper.

## 1 Introduction

When designing a data warehouse, choosing an architecture is crucial. Since it is very dependant on the domain of application and the analysis objectives that are selected for decision support, different solutions are possible. In the ROLAP (Relational OLAP) environment we consider, the most popular solutions are by far star, snowflake, and constellation schemas [7,9], and other modeling possibilities might exist. This choice of architecture is not neutral: it always has advantages and drawbacks and greatly influences the response time of decision support queries. Once the architecture is selected, various optimization techniques such as indexing or materialized views further influence querying and refreshing performance. Again, it is a matter of trade-off between the improvement brought by a given technique and its overhead in terms of maintenance time and additional disk space; and also between different optimization techniques that may cohabit. To help users make these critical choices of architecture and optimization techniques, the performance of the designed data warehouse needs to be assessed. However, evaluating data warehousing and decision support

technologies is an intricate task. Though pertinent, general advice is available, notably on-line [5,12], more quantitative elements regarding sheer performance are scarce. Thus, we propose in this paper a data warehouse benchmark we named DWEB (the *Data Warehouse Engineering Benchmark*). Different goals may be achieved by using a benchmark: (1) compare the performances of various systems in a given set of experimental conditions (users); (2) evaluate the impact of architectural choices or optimisation techniques on the performances of one given system (system designers). The Transaction Processing Performance Council (TPC), a non-profit organization, defines standard benchmarks and publishes objective and verifiable performance evaluations to the industry. Out of the TPC, few decision support benchmarks have been designed. Some do exist, but their specification is not fully published [3]. Some others are not available any more, such as the OLAP APB-1 benchmark that was issued in the late nineties by the OLAP council, an organization whose web site does not exist any more.

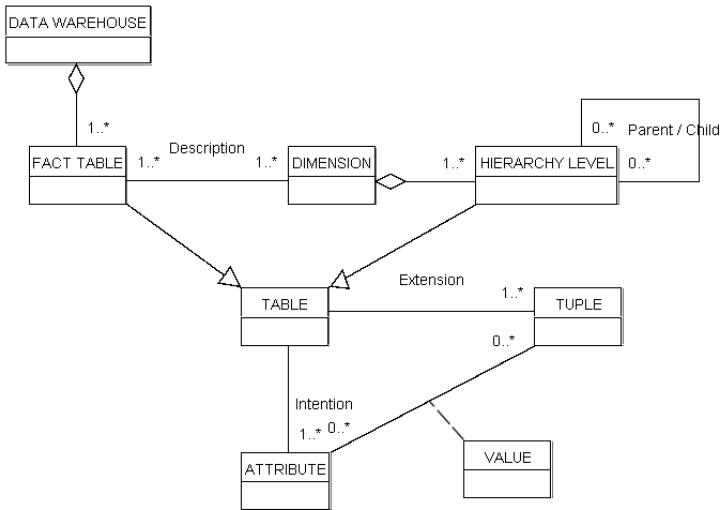
The TPC benchmarks mainly aim at the first benchmarking goal we identified. However, the database schema of TPC benchmarks TPC-H [15] and TPC-R [16] is a classical *product-order-supplier* model, and not a typical data warehouse schema such as a star schema and its derivatives. Furthermore, their workload, though decision-oriented, does not include explicit OLAP (On-Line Analytical Processing) queries either, and they do not address specific warehousing issues such as the ETL (Extract, Transform, Load) process. These benchmarks are indeed implicitly considered obsolete by the TPC that has issued some specifications for their successor: TPC-DS [13]. However, TPC-DS has been under development for three years now and is not completed yet. Furthermore, although the TPC decision support benchmarks are scalable according to Gray's definition [4], their schema is fixed. It must be used "as is". Different ad-hoc configurations are not possible. There is only one parameter to define the database, the Scale Factor ( $SF$ ), which sets up its size (from 1 to 100,000 GB). The user cannot control the size of the dimensions and the fact tables separately, for instance. Finally, the user has no control on the workload's definition. The TPC benchmarks are thus not well adapted to evaluate the impact of architectural choices or optimisation techniques on global performance. For these reasons, we decided to design a full data warehouse synthetic benchmark that would be able to model various ad-hoc configurations of database (modeled as star, snowflake, or constellation schemas) and workload, while being simpler to develop than TPC-DS. We mainly seek to fulfill engineering needs (second benchmarking objective).

This paper presents an overview the DWEB benchmark. First, we present our benchmark's database (metaschema, parameterization, and instantiation into an actual data warehouse) in Section 2. Then, we present the benchmark's workload (query model, parameterization, and workload generation) in Section 3. We illustrate how our benchmark can be used in Section 4 and finally conclude this paper and provide future research directions in Section 5.

## 2 DWEB Database

### 2.1 Schema

Our design objective for DWEB is to be able to model the different kinds of data warehouse architectures that are popular within a ROLAP environment: classical star schemas; snowflake schemas with hierarchical dimensions; and constellation schemas with multiple fact tables and shared dimensions. To achieve this goal, we propose a data warehouse metamodel (represented as a UML class diagram in Figure 1) that can be instantiated into these different schemas. We view this metamodel as a middle ground between the multidimensional metamodel from the Common Warehouse Metamodel (CWM [11]) and the eventual benchmark model. Our metamodel is actually an instance of the CWM metamodel, which could be qualified as a meta-metamodel in our context.



**Fig. 1.** DWEB data warehouse metaschema

Our metamodel is relatively simple, but it is sufficient to model the data warehouse schemas we aim at (star, snowflake, and constellation schemas). Its upper part describes a data warehouse (or a datamart, if a datamart is viewed as a small, dedicated data warehouse) as constituted of one or several fact tables that are each described by several dimensions. Each dimension may also describe several fact tables (shared dimensions). Each dimension may be constituted of one or several hierarchies made of different levels. There can be only one level if the dimension is not a hierarchy. Both fact tables and dimension hierarchy levels are relational tables, which are modeled in the lower part of Figure 1. Classically, a table or relation is defined in intention by its attributes and in extension by its tuples or rows. At the intersection of a given attribute and a given tuple lies the value of this attribute in this tuple.

## 2.2 Parameterization

DWEB’s database parameters help users selecting the data warehouse architecture they need in a given context. The main difficulty in producing a data warehouse schema is parameterizing the instantiation of the metaschema. We indeed try to meet the four key criteria that make a “good” benchmark, as defined by Gray [4]: *relevance*: the benchmark must answer our engineering needs; *portability*: the benchmark must be easy to implement on different systems; *scalability*: it must be possible to benchmark small and large databases, and to scale up the benchmark; and *simplicity*: the benchmark must be understandable, otherwise it will not be credible nor used. Relevance and simplicity are clearly two orthogonal goals. Introducing too few parameters reduces the model’s expressiveness, while introducing too many parameters makes it difficult to apprehend by potential users. Furthermore, few of these parameters are likely to be used in practice. In parallel, the generation complexity of the instantiated schema must be mastered. To solve this dilemma, we propose to divide the parameter set into two subsets. The first subset of so-called low-level parameters allows an advanced user to control everything about the data warehouse generation. However, the number of low-level parameters can increase dramatically when the schema gets larger. For instance, if there are several fact tables, all their characteristics, including dimensions and their own characteristics, must be defined for each fact table. Thus, we designed a layer above with much fewer parameters that may be easily understood and set up (Table 1). More precisely, these high-level parameters are average values for the low-level parameters. At database generation time, the high-level parameters are exploited by random functions (following a gaussian distribution) to automatically set up the low-level parameters. Finally, unlike the number of low-level parameters, the number of high-level parameters always remains constant and reasonable (less than ten parameters). Users may choose to set up either the full set of low-level parameters, or only the high-level parameters, for which we propose default values that correspond to a snowflake schema. These parameters control both schema and data generation.

Note that the cardinal of a fact table is usually lower or equal to the product of its dimensions’ cardinals. This is why we introduce the notion of den-

**Table 1.** DWEB warehouse high-level parameters

Parameter name	Meaning	Def. val.
<i>AVG_NB_FT</i>	Average number of fact tables	1
<i>AVG_NB_DIM</i>	Average number of dimensions per fact table	5
<i>AVG_TOT_NB_DIM</i>	Average total number of dimensions	5
<i>AVG_NB_MEAS</i>	Average number of measures in fact tables	5
<i>AVG_DENSITY</i>	Average density rate in fact tables	0.6
<i>AVG_NB_LEVELS</i>	Average number of hierarchy levels in dimensions	3
<i>AVG_NB_ATT</i>	Average number of attributes in hierarchy levels	5
<i>AVG_HHLEVEL_SIZE</i>	Average number of tuples in highest hierarchy levels	10
<i>DIM_SFCTOR</i>	Average size scale factor within hierarchy levels	10

sity. A density rate of one indicates that all the possible combinations of the dimension primary keys are present in the fact table. When the density rate decreases, we progressively eliminate some of these combinations. This parameter helps controlling the size of the fact table, independently of the size of its dimensions. Furthermore, within a dimension, a given hierarchy level normally has a greater cardinality than the next level. For example, in a *town-region-country* hierarchy, the number of towns must be greater than the number of regions, which must be in turn greater than the number of countries. There is also often a significant scale factor between these cardinalities (e.g., one thousand towns, one hundred regions, ten countries). Hence, we model the cardinality of hierarchy levels by assigning a “starting” cardinality to the highest level in the hierarchy (*HHLEVEL\_SIZE*), and then by multiplying it by a predefined scale factor (*DIM\_SFACTOR*) for each lower-level hierarchy. Finally, since some of DWEB’s parameters might sound abstract, the data warehouse global size (in megabytes) is assessed at generation time so that users retain full control over it and may adjust the parameters to better represent the kind of warehouse they need.

### 2.3 Generation Algorithm

The instantiation of the DWEB metaschema into an actual benchmark schema is done in two steps: (1) build the dimensions; (2) build the fact tables. Due to space constraints, the pseudo-code for these two steps is not provided here, but it is available on-line [2]. Each of these steps is further subdivided, for each dimension and each fact table, into generating its intention and extension. In addition, hierarchies of dimensions are managed.

## 3 DWEB Workload

In a data warehouse benchmark, the workload may be subdivided into a load of decision support queries (mostly OLAP queries) and the ETL (data generation and maintenance) process. To design DWEB’s workload, we inspire both from TPC-DS’ workload definition and information regarding data warehouse performance from other sources [1,6]. However, TPC-DS’ workload is very elaborate and sometimes confusing. Its reporting, ad-hoc decision support and OLAP query classes are very similar, for instance, but none of them include any specific OLAP operator such as Cube or Rollup. Since we want to meet Gray’s simplicity criterion, we propose a simpler workload. Furthermore, we also have to design a workload that is consistent with the variable nature of the DWEB data warehouses. We also, in a first step, mainly focus on the definition of a query model. Modeling the full ETL process is a complex task that we postpone for now. We consider that the current DWEB specifications provide a raw loading evaluation framework. The DWEB database may indeed be generated into flat files, and then loaded into a data warehouse using the ETL tools provided by the system.

### 3.1 Query Model

The DWEB workload models two different classes of queries: purely decision-oriented queries involving common OLAP operations, such as cube, roll-up, drill down and slice and dice; and extraction queries (simple join queries). We define our generic query model as a grammar that is a subset of the SQL-99 standard. Due to space constraints, this query model is only available on-line [2].

### 3.2 Parameterization

DWEB’s workload parameters help users tailoring the benchmark’s load, which is also dependent from the warehouse schema, to their needs. Just like DWEB’s database parameter set, DWEB’s workload parameter set (Table 2) has been designed with Gray’s simplicity criterion in mind. These parameters determine how the query model is instantiated. These parameters help defining the workload’s size and complexity, by setting up the proportion of complex OLAP queries (i.e., the class of queries) in the workload, the number of aggregation operations, the presence of a Having clause in the query, or the number of subsequent drill down operations. Here, we have only a limited number of high-level parameters. Indeed, it cannot be envisaged to dive further into detail if the workload is as large as several hundred queries, which is quite typical. Note that  $NB\_Q$  is only an *approximate* number of queries because the number of drill down operations after an OLAP query may vary. Hence we can stop generating queries only when we actually have generated as many or more queries than  $NB\_Q$ .

**Table 2.** DWEB workload parameters

Parameter name	Meaning	Def. val.
$NB\_Q$	Approximate number of queries in the workload	100
$AVG\_NB\_ATT$	Average number of selected attributes in a query	5
$AVG\_NB\_RESTR$	Average number of restrictions in the query	3
$PROB\_OLAP$	Probability that the query type is OLAP	0.9
$PROB\_EXTRACT$	Probability that the query is an extraction query	$1 - P\_OLAP$
$AVG\_NB\_AGGREG$	Average number of aggregations in an OLAP query	3
$PROB\_CUBE$	Probability of an OLAP query to use the Cube operator	0.3
$PROB\_ROLLUP$	Probability of an OLAP query to use the Rollup operator	$1 - P\_CUBE$
$PROB\_HAVING$	Probability of an OLAP query to include an Having clause	0.2
$AVG\_NB\_DD$	Average number of drill downs after an OLAP query	3

### 3.3 Generation Algorithm

Due to space constraints, the pseudo-code of DWEB’s workload generation algorithm is only available on-line [2]. However, its principle follows. The algorithm’s purpose is to generate a set of SQL-99 queries that can be directly executed on the synthetic data warehouse defined in Section 2. It is subdivided into two steps: (1) generate an initial query that may either be an OLAP or an extraction (join)



query; (2) if the initial query is an OLAP query, execute a certain number of drill down operations based on the first OLAP query. More precisely, each time a drill down is performed, an attribute from a lower level of dimension hierarchy is added to the attribute clause of the previous query. Step 1 is further subdivided into three substeps: (1) the Select, From, and Where clauses of a query are generated simultaneously by randomly selecting a fact table and dimensions, including a hierarchy level within a given dimension hierarchy; (2) the Where clause is supplemented with additional conditions; (3) eventually, it is decided whether the query is an OLAP query or an extraction query. In the second case, the query is complete. In the first case, aggregate functions applied to measures of the fact table are added in the query, as well as a Group by clause that may include either the Cube or the Rollup operator. A Having clause may optionally be added in too. The aggregate function we apply on measures is always Sum since it is the most common aggregate in cubes. Furthermore, other aggregate functions bear similar time complexities, so they would not bring in any more insight in a performance study.

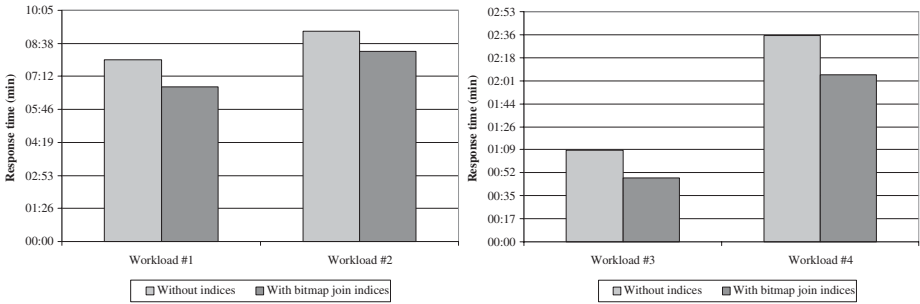
## 4 Sample Usage of DWEB

In order to illustrate one possible usage for DWEB, we tested the efficiency of bitmap join indices, which are well suited to the data warehouse environment, on decision support queries under Oracle. The aim of this particular example is to compare the execution time of a given workload on a given data warehouse, with and without using bitmap join indices.

First, we generated a data warehouse modeled as a snowflake schema. This schema is organized around one fact table that is described by five dimensions, each bearing two to three hierarchy levels. The fact table contains about 140,000 tuples, the dimension hierarchy levels about ten tuples on an average, for a global size of about 4 MB (this is a voluntarily small example and not a full-scale test). We applied different workloads on this data warehouse. *Workload#1* is a typical DWEB workload constituted of fifty queries. 10% of these queries are extraction (join) queries and the rest are decision support queries involving OLAP operators (Cube and Rollup). In *Workload#1*, we limited the queries to the dimensions' lowest hierarchy levels, i.e., to the star schema constituted of the fact table and the "closest" hierarchy levels. *Workload#2* is similar to *Workload#1*, but it is extended with drill down operations that scan the dimensions' full hierarchies (from the highest level to the lowest level). Thus, this workload exploits the whole snowflake schema. To evaluate the efficiency of bitmap join indices, we timed the execution of these two workloads on our test data warehouse (response time is our only performance metric for now), first with no index, and then by forcing the use of five bitmap join indices defined on the five dimensions (for the lowest hierarchy levels in *Workload#1* and for the whole hierarchies in *Workload#2*). To flatten any response time variation in these experiments, we replicated each test ten times and computed the average response times. We made sure *a posteriori* that the standard deviation was close to zero. These tests have been executed on a

PC with a Celeron 900 processor, 128 MB of RAM, an IDE hard drive, and running Windows XP Professional and Oracle 9i.

The left-hand graph on Figure 2 represents the average response time achieved for *Workload#1* and *#2*, with and without bitmap join indices, respectively. It shows a gain in performance of 15% for *Workload#1*, and 9.4% for *Workload#2*. This decrease in efficiency was expected, since the drill down operations added in *Workload#2* are costly and need to access the data (bitmap join indices alone cannot answer such queries). However, the overall performance improvement we achieved was not as good as we expected. We formulated the hypothesis that the extraction queries, which are costly joins and need to access the data too, were not fully benefiting from the bitmap join indices. To confirm this hypothesis, we generated two new workloads, *Workload #3* and *#4*. They are actually almost identical to *Workload#1* and *Workload#2*, respectively, but do not include any extraction (join) queries. Then, we repeated our experiment following the same protocol. The right-hand graph on Figure 2 represents the average response time achieved for *Workload#3* and *#4*, with and without bitmap join indices, respectively. This time, we obtained similar results than in our previous experiment (in trend): response time clearly increases when drill down operations are included into the workload. However, response time is now much better and the gain in performance is 30.9% for *Workload#3*, and 19.2% for *Workload#4*.



**Fig. 2.** Test results

In conclusion, we want to point out that these experiments are not very significant *per se*, and do not do justice to Oracle. However, we illustrated how DWEB could be used for performance evaluation purposes. These experiments could also be seen as a (very basic) performance comparison between two different data warehouse architectures (star schema and snowflake schema). Our results indeed conform to the well-known fact that introducing hierarchies into a star schema induces more join operations in the decision support queries, and hence degrade their response time. Finally, we were also able to witness the impact of costly join operations on a data warehouse structure that is not properly indexed to answer such queries.

## 5 Conclusion and Perspectives

We proposed in this paper a new data warehouse benchmark called DWEB (the *Data Warehouse Engineering Benchmark*) that is aimed at helping data warehouse designers to choose between alternate warehouse architectures and performance optimization techniques. When designing DWEB, we tried to grant it the characteristics that make up a “good” benchmark according to Gray: relevance, portability, scalability, and simplicity. To make DWEB relevant for evaluating the performance of data warehouses in an engineering context, we designed it to generate different data warehouse schemas (namely star, snowflake and constellation schemas) and workloads. Note that the database schema of TPC-DS, the future standard data warehouse benchmark currently developed by the TPC, can be modeled with DWEB. In addition, though DWEB’s workload is not currently as elaborate as TPC-DS’s, it is also much easier to implement. It will be important to fully include the ETL process into our workload, though, and the specifications of TPCD-DS and some other existing studies [10] might help us. We now need to further test DWEB’s relevance on real cases. To achieve this goal, we plan to compare the efficiency of various index and materialized view selection techniques. We also made DWEB very tuneable to reach both the relevance and scalability objectives. However, too many parameters make the benchmark complex to use and contradict the simplicity requirement. Though it is impossible to achieve both a high simplicity and a high relevance and scalability, we introduced a layer of high-level parameters that are both simpler than the potentially numerous low-level parameters, and in reduced and constant number. DWEB might not be qualified as a simple benchmark, but our objective was to keep its complexity as low as possible. Finally, portability was achieved through a Java implementation. DWEB’s latest version is freely available on-line [8]. Finally, we also illustrated with a practical case how DWEB can be used.

This work opens up many perspectives for developing and enhancing DWEB. In this paper, we assumed an execution protocol and performance metrics were easy to define for DWEB (e.g., using TPC-DS’ as a base) and focused on the benchmark’s database and workload model. A more elaborate execution protocol must be designed, especially since two executions of DWEB using the same parameters produce different data warehouses and workloads. This is interesting when, for instance, one optimization technique needs to be tested against many databases. However, note that it is also possible to save a given warehouse and its associated workload to run tests on different systems and/or with various optimization techniques. Defining sound metrics (beside response time) would also improve DWEB’s usefulness. In this area, we could inspire from metrics designed to measure the quality of data warehouse conceptual models [14]. We are also currently working on warehousing complex, non-standard data (including multimedia data, for instance). Such data may be stored as XML documents. Thus, we also plan a “complex data” extension of DWEB that would take into account the advances in XML warehousing. Finally, more experiments with DWEB should also help us propose sounder default parameter values. We also encourage other people to report on their own experiments.

## References

1. BMC Software. Performance Management of a Data Warehouse. <http://www.bmc.com>, 2000.
2. J. Darmont, F. Bentayeb, and O. Boussaïd. The Design of DWEB. Technical report, ERIC, University of Lyon 2, France, June 2005. <http://eric.univ-lyon2.fr/~jdarmont/publications/files/dweb.pdf>.
3. M. Demarest. A Data Warehouse Evaluation Model. *Oracle Technical Journal*, 1(1):29, October 1995.
4. J. Gray. *The Benchmark Handbook for Database and Transaction Processing Systems*. Morgan Kaufmann, second edition, 1993.
5. L. Greenfield. Performing Data Warehouse Software Evaluations. <http://www.dwinfocenter.org/evals.html>, 2004.
6. L. Greenfield. What to Learn About in Order to Speed Up Data Warehouse Querying. <http://www.dwinfocenter.org/fstquery.html>, 2004.
7. W. Inmon. *Building the Data Warehouse*. John Wiley & Sons, third edition, 2002.
8. B. Joubert and S. Guesmoa. DWEB Java prototype v0.31. <http://bdd.univ-lyon2.fr/download/dweb.tgz>, 2005.
9. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, second edition, 2002.
10. A. Labrinidis and N. Roussopoulos. A performance evaluation of online warehouse update algorithms. Technical Report CS-TR-3954, Department of Computer Science, University of Maryland, November 1998.
11. Object Management Group. *Common Warehouse Metamodel (CWM) Specification version 1.1*, March 2003.
12. N. Pendse. The OLAP Report: How not to buy an OLAP product. [http://www.olapreport.com/How\\_not\\_to\\_buy.htm](http://www.olapreport.com/How_not_to_buy.htm), December 2003.
13. M. Poess, B. Smith, L. Kollar, and P.-A. Larson. TPC-DS: Taking Decision Support Benchmarking to the Next Level. In *ACM SIGMOD 2002, Madison, USA*, June 2002.
14. M. Serrano, C. Calero, J. Trujillo, S. Luján-Mora, and M. Piattini. Empirical Validation of Metrics for Conceptual Models of Data Warehouses. In *16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Riga, Latvia*, volume 3084 of *LNCS*, pages 506–520, 2004.
15. Transaction Processing Performance Council. *TPC Benchmark H Standard Specification version 2.1.0*, August 2003.
16. Transaction Processing Performance Council. *TPC Benchmark R Standard Specification version 2.1.0*, August 2003.

# A Set of Quality Indicators and Their Corresponding Metrics for Conceptual Models of Data Warehouses

Gema Berenguer<sup>1</sup>, Rafael Romero<sup>1</sup>, Juan Trujillo<sup>1</sup>, Manuel Serrano<sup>2</sup>,  
and Mario Piattini<sup>2</sup>

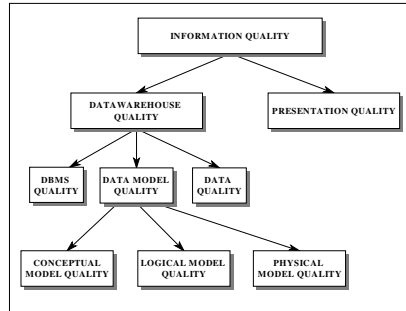
<sup>1</sup> Dept. de Lenguajes y Sistemas Informáticos,  
Universidad de Alicante,  
Apto. Correos 99. E-03080. Alicante.  
{gberenguer, romero, jtrujillo}@dlsi.ua.es  
<sup>2</sup> Escuela Superior de Informática,  
University of Castilla – La Mancha,  
Paseo de la Universidad, 4, 13071 Ciudad Real  
{Manuel.Serrano, Mario.Piattini}@uclm.es

**Abstract.** The quality of Data Warehouses is absolutely relevant for organizations in the decision making process. The sooner we can deal with quality metrics (i.e. conceptual modelling), the more willing we are in achieving a data warehouse (DW) of a high quality. From our point of view, there is a lack of more objective indicators (metrics) to guide the designer in accomplishing an outstanding model that allows us to guarantee the quality of these data warehouses. However, in some cases, the goals and purposes of the proposed metrics are not very clear on their own. Lately, quality indicators have been proposed to properly define the goals of a measurement process and group quality measures in a coherent way. In this paper, we present a framework to design metrics in which each metric is part of a quality indicator we wish to measure. In this way, our method allows us to define metrics (theoretically validated) that are valid and perfectly measure our goals as they are defined together a set of well defined quality indicators.

**Keywords:** Quality indicators, quality metrics, conceptual modelling, data warehouses, multidimensional modelling

## 1 Introduction

Data Warehouses (DWs), which are the core of current decision support systems, provide companies with many years of historical information for the decision making process [10]. The term data warehouse is defined as “a subject-oriented, integrated, time-variant, non-volatile collection of data supporting management’s decisions” [8]. A lack of quality in the data warehouse can be disastrous consequences from both a technical and organizational point of view. Therefore, it is crucial for an organization to guarantee the quality of the information contained in these DWs. The information quality of a DW is determined by (i) the quality of the DBMS (Database Management System), (ii) the quality of the data models used in their design, (iii) the quality of the data themselves contained in the data warehouse (see figure 1).



**Fig. 1.** Quality of the information and the data warehouse

In order to guarantee the quality of the DBMS, we can use an International Standard such as ISO/IEC 9126 [9] or one of the comparative studies of existing products. The quality of the datawarehouse model also strongly influences information quality. The model can be considered at three levels: conceptual, logical and physical. Due to space constraints, we refer the reader to [1] for a deep comparison of conceptual, logical and physical models proposed for data warehouses. At the logical level several recommendations exist in order to create a good dimensional data model [11] and in recent years we have proposed and validated both theoretically and empirically several metrics that enable the evaluation of the complexity of star models. At the physical model depends on each system and consist of selecting the physical tables, indexes, data partitions, etc. [2] [11].

However, from our point of view, we claim that design guidelines or subjective quality criteria are not enough to guarantee the quality of multidimensional models for DWs. Therefore, we believe that a set of formal and quantitative measures should be provided to reduce subjectivity and bias in evaluation, and guide the designer in his work. However, we cannot assure that quality measures interpret a measurable concept on their own with guarantee. So, lately, quality indicators have been proposed to define the concept to be measured and group the quality measures needed to measure that indicator [7]. Otherwise, we may propose metrics that cannot measure what they are defined for and they may overlap the measurable concept.

In this paper, we firstly propose a set of quality indicators to measure the quality of conceptual schemas for DWs. Then, once these indicators clearly establish the set of concepts to be measured, we define the set of the corresponding quality metrics that will measure that indicator. On defining these indicators and quality metrics, we use our conceptual modelling approach, based on the Unified Modelling Language (UML), to properly accomplish the conceptual modelling of data warehouses [13]. In this paper, we have focused in the first step of the conceptual modelling of DWs and we will use the package diagrams to model complex and huge DWs thereby facilitating their modelling and understanding [13]. Then, we use our quality indicators and measures to an example to show the benefit of our proposal. Due to space constraints, we cannot provide the theoretical validation we have accomplished using both the (i) axiomatic approach and (ii) the measure theory.

The rest of the paper is structured as follow: section 2 presents the method we follow for defining and obtaining correct quality indicators and metrics. Section 3

presents a summary of UML package diagrams we use in this paper for the conceptual modelling of data warehouses. In Section 4, we define the proposed quality indicators and metrics and present some examples to show how to apply them. Finally, section 5 draws conclusions and immediate future works arising from the conclusions reached in this work.

## 2 A Method to Define Quality Indicators and Metrics

A measurable concept is an abstract relation between attributes of one or more entities, and a necessity of information. Some examples of measurable concepts are: quality, reliability, accessibility and so on. Metrics cannot interpret on their own a measurable concept, and therefore, it is essential to use quality indicators [7]. A metric assess a characteristic of an object while an indicator will use one or more metrics to measure something. Thus, indicators are the basis for (i) quantify measurable concepts for a necessity of information, (ii) quantitative methods of evaluation or prediction, and (iii) to provide information to take decisions.

The definition of quality indicators and metrics has to be accomplished in a methodological way, which makes necessary to accomplish a set of stages to be able to assure their liability. Next, we will present a modification of the methods proposed in [5] to define quality metrics, and the method proposed in MMLC (Measure Model Life Cycle) [6]; to allow us to incorporate the definition of quality indicators and metrics in an overall approach.

This method can be structured into three main phases: (i) creating the indicator, (ii) defining the required metrics for the indicator and (iii) applying these metrics to measure a conceptual schema. In the first phase, we have to find the main objective that we pursue, and then, define the corresponding indicator to achieve that objective. Next, in the second phase, we define the list of required metrics that will allow us to measure the indicator. On creating a metric, we will firstly define it, and then, we have to accomplish the theoretical and empirical validation [5]. At the end of this paper, we will present a summary of the frameworks we use for the theoretical validation of our metrics.

To accomplish the empirical validation of metrics, we need to set a family of experiments [5], from which we will obtain a set of thresholds that will be later applied to the indicator algorithm. Once the metric has been properly defined, we pass to the third phase by applying the obtained metrics to a conceptual schema. With the valid metrics and the thresholds obtained from the empirical validation, we will define the algorithm to measure the indicator. Finally, after analyzing the results obtained by the indicator algorithm, we will store and communicate these results.

### 2.1 Indicator Template

There are some organizations that do not achieve the expected benefits of applying quality indicators due to the fact that these quality indicators have not been properly specified or they are not properly interpreted [7]. Therefore, we will document the specification of indicators, their interpretation and use as proposed in [7], in order to avoid inconsistencies in their definitions.

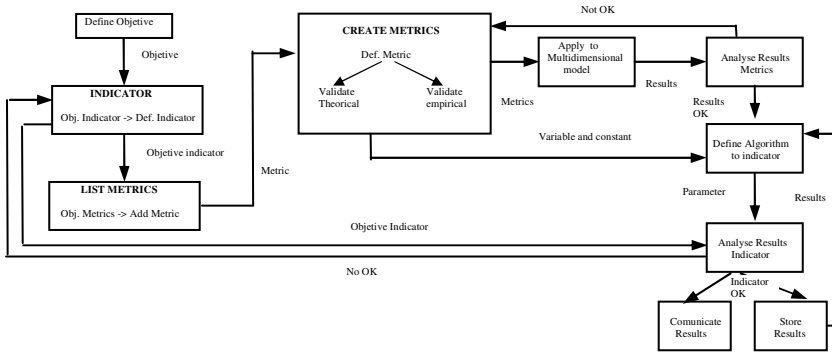


Fig. 2. Method for defining quality indicators and metrics

The Software Engineering Institute (SEI) has found that an indicator template can help an organization to improve its software measurement processes and infrastructure [7]. In this work, authors describe a template that can be used to precisely describe, document, and report *who, what, when, where, why, and how* to define organization’s indicators. Moreover, they also describe the use of the indicator template within the context of the Goal-Driven Software Measurement (GQ(I)M) methodology and the Capability Maturity Model Integration (CMMI) framework.

Therefore, due to the high importance of quality indicators in our proposal, in the following, we will present the indicator template we have followed – proposed in [7]. Thus, our indicator template consists of:

- *Indicator objective*: the objective or purpose of the indicator
- *Questions*: the questions that the user of the indicator is trying to answer
- *Visual display*: a graphical view of the indicator
- *Perspective or viewpoint*: the description of the audience for whom the indicator is intended
- *Inputs*: the list of the measures required to construct the indicator and its definitions
- *Algorithms*: the description of the algorithm used to construct the indicator from the measures
- *Assumptions*: the list of assumptions about the organization, its processes, life-cycle model, and so on that are important conditions for collecting and using the indicator.
- *Data collection information*: information pertaining to how, when, how often, by whom, etc. the data elements required to construct the indicator are collected.
- *Data reporting information*: information on who is responsible for reporting the data, to whom, and how often.
- *Data storage*: information on storage, retrieval, and security of the data.
- *Analysis and interpretation of results*: information on how to analyze and interpret as well as to not misinterpret the indicator.

At this point, we have stated the reason why we use quality indicators and the corresponding template use to define them. Thus, in Table 1, we match each relevant step of our method (see Figure 2) with the corresponding indicator template issue.



**Table 1.** Correspondence between our indicator template issues and method phases

Indicator Template	Phase Method
Indicator Name / Title	Indicator
Objective	Indicator
Questions	Indicator
Visual Display	Communicate results
Inputs	Create metrics
Data Collection	Apply multidimensional model
Data Reporting	Communicate results
Data Storage	Store results
Algorithm	Define algorithm indicator
Interpretation	Store results
Analysis	Analyze Data

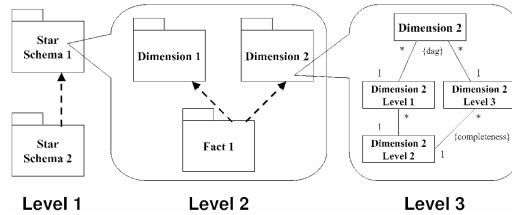
### 3 Multidimensional Modelling with Package Diagrams of UML

In previous works, we have proposed a DW development method [12], based on the Unified Modelling Language (UML) and the Unified Process (UP), to properly design all aspects of a DW. More specifically, we have dealt with the modelling of different aspects of a DW by using the UML: MD modelling [13] (i.e. the aim of this paper), modelling of the ETL processes, modelling data mappings between data sources and targets [12], modelling physical aspects of DWs at the conceptual level etc. In this section, we outline our approach of using UML package diagrams for the conceptual modelling of large data warehouses [13], which is the approach in which we based on in this paper for the definition of quality indicators and metrics. Based on our experience in real-world cases, we have developed a set of design guidelines for using UML packages in MD modelling. Our approach proposes the use of UML packages in order to group classes together into higher level units creating different levels of abstraction, and therefore, simplifying the final multidimensional (MD) model. In this way, when modelling complex and large DW systems, the designer is not restricted to use flat UML class diagrams. We refer to [13] for a complete description of all design guidelines we have defined.

In Figure 3, we summarize the three main levels in which we structure the conceptual modelling of large data warehouses. At level 1, we define one package for each different star schema we consider in our design and we call them star package. A dependency between two packages at this level represents that they share at least one dimension or one fact. Then, at level 2 we define one package for each dimension and fact considered in our design and we call them *dimension package* and *fact package*, respectively. There is always a dependency between the fact package and the dimension packages meaning that one fact consists on the corresponding dimensions. A dependency between two dimension packages means that they share at least one classification hierarchy level. Finally, at level 3, we specify the whole content of both dimension and fact packages. As seen in this Figure 3, at level 3, each

<sup>1</sup> Although star schema is a logical schema, we refer to star schema to the abstract definition of one fact and several dimensions.

dimension package will contain the definition of the corresponding dimension and their classification hierarchy levels. We should notice that the dependencies between packages allow us to define one element (package, fact or dimension) just once in our design and then re-utilise it whenever convenient.



**Fig. 3.** The three levels of our MD modelling approach with UML package diagrams

Our whole approach for the conceptual modelling of DWs has been specified by means of a UML profile that contains the necessary stereotypes in order to carry out conceptual modelling successfully. Due to space constraints, we refer the reader to [13] for further details on the profile.

## 4 Quality Indicators and Metrics

Prior to the definition of an indicator we must clearly and precisely know the goal of what we want to measure. The structural properties such as the structural complexity of a schema have an impact on its cognitive complexity [4] and on the mental burden of the persons who have to deal with the artefact. High cognitive complexity leads an artefact to reduce their understandability, analyzability and modifiability. Leading to undesirable external quality attributes [9] [4]. For this reason, it is desirable that a schema has excellent structural properties to be able to achieve good quality. In this paper, our goal will be to minimize the structural complexity of the conceptual schemas to guarantee their quality.

Once the main goal has been set, we have to define the corresponding indicator to measure it. As in this paper, we work with levels 1 and 2 of our package diagram proposal (see Figure 2), we need to define one indicator for each level. If we are able to obtain the minimum structural complexity in both levels, we will therefore obtain the minimum structural complexity in the final conceptual schema.

After having defined the indicators, we must establish the elements we need to measure to further define the corresponding metrics to measure them:

*Number of input and output relationships per package.*

*Number of input and output relationships between two packages*

*Number of output relationships of a package with regard to the total relationships that exist on the model.*

Thus, we will proceed with the definition of the required metrics. These metrics will be applied at level 1 (diagram) and 2 (package) of our approach (Figure 2).

**Table 2.** Diagram level metrics

Metric	Description
NP(S)	Number of packages of the diagram S
NRES1(S)	Number of input and output relationships of the diagram level
NRESP(S)	Number of input and output relationships between two packages
RESP(S)	Ratio of input and output relationships per number of packages $RESP(S) = NRES1(S) / NP(S)$

**Table 3.** Package level metrics

Metric	Description
NRS(P)	Number of output relationships of a package P
RST(P)	Ratio of output relationships of a package P by the total relationships of this package $RST(P) = NRS(P) / NRES1(S)$

As one of the goals was to obtain the minimum complexity of diagrams at level 2, we define an indicator to measure the structural complexity of diagrams at this level. On defining the indicator, the next step is to know what we need to measure:

*Number of output relationships of a dimension package with regard to the total input and output relationships of this package.*

*Number of input and output relationships between two packages.*

*Number input and output relationships between two dimension packages by the number of dimension packages that exists.*

In tables 4 and 5 we can find the metrics we have defined:

**Table 4.** Package level metrics

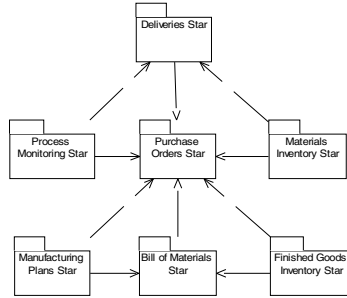
Metric	Description
NREDP(P)	Number of input relationships to a package dimension P
NRSDP(P)	Number of output relationships of a package dimension P
RSDT(P)	Ratio of relationships out of a dimension package P with regard to the total number of input and output relationships to this package $RSDT(P) = NRSDP(P) / (NREDP(P) + NRSDP(P))$

**Table 5.** Diagram level metrics

Metric	Description
NIDP(S)	Number of dimension packages imported from another diagrams
NDDP(S)	Number of dimension packages defined in the diagram
NDP(S)	Number total of packages of the diagram S $NDP(S) = NIDP(S) + NDDP(S) + 1$
NRTDP(S)	Number of input and output relationships between dimension packages
NRESDP(S)	Number of input and output relationships between two dimension packages
RDP(S)	Ratio of input and output relationships between dimension packages by the number of the dimension packages. $RDP(S) = NRTDP(S) / (NDP(S) - 1)$

**4.1 Example**

In this section we apply the previously-defined metrics to an example. We have applied our package diagram approach to a supply value chain example completely developed in [13]. In Figure 4, we show the level 1 of the model that is composed



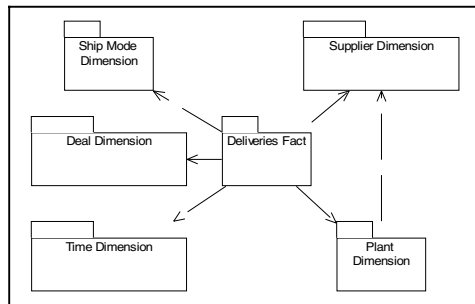
**Fig. 4.** Level 1: different star schemas of the supply value chain example

**Table 6.** Level (level 1) package metrics

	NRS	RST
Deliveries	1	1/10
Process Monitoring	2	2/10
Purchase Orders	0	0/10
Materials Inventory	2	2/10
Manufacturing Plans	2	2/10
Bill of Materials	1	1/10
Finished Inventory	2	2/10

**Table 7.** Metric NERSP<sup>2</sup>

NERSP	Value
Deliveries – Process Monitoring	1
Deliveries – Purchase Orders	1
Deliveries – Materials Inventory	1
Purchase Orders – Materials Inventory	1
Purchase Orders – Process Monitoring	1
Purchase Orders – Manufacturing Plans	1
Purchase Orders –Bill of materials	1
Purchase Orders – Finished Inventory	1
Manufacturing Plans – Bill of materials	1
Bill of materials – Finished inventory	1



**Fig. 5.** Level 2: Deliveries Star

<sup>2</sup> Only represent the metrics that value NERSP is different zero.

by seven packages that represent the different star schemas. Then, in Tables 6, 7 and 8 we present the obtained values for the proposed metrics.

In Figure 5, we show the content of the package *Deliveries Star* (level 2). Tables 9, 10 and 11 show the values for the proposed metrics.

The theoretical validation helps us to know when and how apply the metrics. There are two main tendencies in metrics validation: the frameworks based on axiomatic approaches [15] [3] and the ones based on the measurement theory [14][16]. We have validated our metrics by using both frameworks. However, due to space constraints, we cannot provide these theoretical validations in this paper.

**Table 8.** Level (level 1) diagram metrics

	Value
NP(S)	7
NRES1(S)	10
RESP(S)	10/7

**Table 9.** Metric NERSP<sup>3</sup>

NERSP	Value
Plant - Supplier	1

**Table 10.** Level (level 2) package metrics

	NREPD	NRSPD	RSDT
Ship Mode	1	0	0/1
Deal	1	0	0/1
Time	1	0	0/1
Plant	1	1	1/2
Supplier	2	0	0/2

**Table 11.** Level (level 2) diagram metrics

	Value
NIDP	3
NDDP	2
NDP	6
NRTDP	1
RDP(S)	1/5

## 6 Conclusions

In this paper we have focused on the quality of the conceptual models of data warehouses. We have mainly focused on those models that use UML packages to model data warehouses. We have proposed a set of quality indicators and the metrics on which they are based on in order to assure the quality of the data warehouses conceptual models. These quality indicators have allowed us to clearly define quantifiable elements in which we based on for measuring the quality of the models. In order to obtain high confidence indicators, we have defined the metrics for each indicator we have defined.

Those metrics have been theoretically validated using two formal frameworks, each of them representing a validating approach: axiomatic approaches and those approaches based on measurement theory. This paper has presented the first steps in obtaining a valid set of quality indicators. We are now focusing on develop the empirical validation with the proposed indicators and metrics in order to obtain a valid and useful set of quality indicators for data warehouse conceptual models.

<sup>3</sup> Only represent the metrics that value NERSP is different zero.

## References

1. Abelló, A., Samos, J. and Saltor, F. *YAM2 (Yet Another Multidimensional Model): An extension of UML*. in *International Database Engineering & Applications Symposium (IDEAS'02)*. 2002.
2. Bouzeghoub, M. and Kedad, Z., *Quality in Data Warehousing*, in *Information and database quality*. 2002, Kluwer Academic Publishers.
3. Briand, L., Morasca, S. and Basili, V., *Property-Based Software Engineering Measurement*. IEEE Transactions on Software Engineering, 1996. **22(1)**: p. 68-86.
4. Briand, L., Wüst, J. and Lounis, H. *A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: an Industrial Case Study*. in *21st International Conference on Software Engineering*. 1999. Los Angeles, California.
5. Calero, C., Piattini, M. and Genero, M. *Method for obtaining correct metrics*. in *3<sup>rd</sup> International Conference on Enterprise and Information Systems (ICEIS 2001)*. 2001.
6. Cantone, G. and Donzelli, P., *Production and maintenance of software measurement models*. Journal of Software Engineering and Knowledge Engineering, 2000. **5**: p. 605-626.
7. Goethert, W. and Siviyy, J., *Applications of the Indicator Template for Measurement and Analysis Initiative, Technical Note CMU/SEI-2004-TN-024*. 2004.
8. Inmon, W. H., *Building the Data Warehouse*. 3rd Edition ed. 2002, USA: John Wiley and Sons.
9. ISO/IEC, *9126-1: Software Engineering - Product quality - Part 1: Quality model*. 2001.
10. Jarke, M., Lenzerini, M., Vassiliou, Y. and Vassiliadis, P., *Fundamentals of Data Warehouses*. second edition ed. 2002: Springer-Verlag.
11. Kimball, R. and Ross, M., *The Data Warehouse Toolkit, 2<sup>nd</sup> edition*. 2002: John Wiley & Sons.
12. Luján-Mora, S. and Trujillo, J. *A Data Warehouse Engineering Process*. in *3rd International Conference in Advances in Information Systems(ADVIS2004)*. 2004.Izmir (Turkey).
13. Luján-Mora, S., Trujillo, J. and Song, I.-Y. *Multidimensional Modeling with UML Package Diagrams*. in *21st International Conference on Conceptual Modeling, LNCS 2503*. 2002.
14. Poels, G. and Dedene, G., *Distance-based software measurement: necessary and sufficient properties for software measures*. Information and Software Technology, 2000. **42(1)**: p. 35-46.
15. Weyuker, E., *Evaluating Software Complexity Measures*. IEEE Transactions on Software Engineering, 1988. **14(9)**: p. 1357-1365.
16. Zuse, H., *A Framework of Software Measurement*. 1998, Berlin: Walter de Gruyter.

# Design and Development of a Tool for Integrating Heterogeneous Data Warehouses

Riccardo Torlone and Ivan Panella

Dipartimento di Informatica e Automazione,  
Università degli studi Roma Tre  
torlone@dia.uniroma3.it

**Abstract.** In this paper we describe the design of a tool supporting the integration of independently developed data warehouses, a problem that arises in several common scenarios. The basic facility of the tool is a test of the validity of a matching between heterogeneous dimensions, according to a number of desirable properties. Two strategies are then provided to perform the actual integration. The first approach refers to a scenario of loosely coupled integration, in which we just need to identify the common information between sources and perform drill-across queries over them. The goal of the second approach is the derivation of a materialized view built by merging the sources, and refers to a scenario of tightly coupled integration in which queries are performed against the view. We illustrate architecture and functionality of the tool and the underlying techniques that implement the two integration strategies.

## 1 Introduction

Today, a common practice for building a data warehouse is to develop a series of individual data marts, each of which provides a dimensional view of a single business process. These data marts should be based on common dimensions and facts, but very often they are developed independently, and it turns out that their integration is a difficult task. Indeed, the need for combining autonomous (i.e., independently developed and operated) data marts arises in other common cases. For instance, when different companies get involved in a federated project or when there is the need to combine a proprietary data warehouse with external data, perhaps wrapped from the Web.

We have studied this problem from a conceptual point of view, by introducing the notion of *dimension compatibility* underlying data mart integration [4], which extends an earlier notion proposed by Kimball [7]. Intuitively, two dimensions (belonging to different data marts) are compatible if their common information is consistent. Building on this study, in this paper we illustrate the design of a practical integration tool for multidimensional databases, similar in spirit to other tools supporting heterogeneous data transformation and integration [9].

The basic facility of the tool is the integration of a pair of autonomous dimensions. We have identified a number of desirable properties that a *matching* between dimensions (that is, a one-to-one correspondence between their levels)

should satisfy: the *coherence* of the hierarchies on levels, the *soundness* of the paired levels, according to the members associated with them, and the *consistency* of the functions that relate members of different levels within the matched dimensions. The tool makes use of a powerful technique, the *chase of dimensions*, to test for these properties.

Two different integration strategies are supported by the system. The first one refers to a scenario of loosely coupled integration, in which we need to identify the common information between sources (intuitively, the intersection), while preserving their autonomy. This approach supports *drill-across queries* [7], based on joining data over common dimension, to be performed over the original sources. The goal of the second approach is rather merging the sources (intuitively, making the union) and refers to a scenario of tightly coupled integration, in which we need to build a materialized view that embeds the sources. Under this approach, queries are performed against the view built from the sources.

The integration of heterogenous databases has been studied in the literature extensively (see, for instance, [5,8,12]). In this paper, we take apart the general aspects of the problem and concentrate our attention on the specific problem of integrating *multidimensional* data. Differently from the general case, this problem can be tackled in a more systematic way for two main reasons. First, multidimensional databases are structured in a rather uniform way, along the widely accepted notions of dimension and fact. Second, data quality in data warehouses is usually higher than in generic databases, since they are obtained by reconciling several data sources. To our knowledge, the present study is the first systematic approach to this problem. Some work has been done on the problem of integrating data marts with external data, stored in various formats: XML [6,10] and object-oriented [11]. This is related to our tightly coupled approach to integration, in that dimensions are “enriched” with external data. Moreover, our loosely coupled approach to integration is related to the problem of drill-across [1]. However, the goal of these studies is different from ours.

The rest of the paper is organized as follows. In Section 2 we provide the basic notions underlying our approach. In Section 3 we illustrate the techniques for dimension integration and describe how they can be used to integrate autonomous data marts. In Section 4 we present architecture and functionality of the tool and finally, in Section 5, we sketch some conclusions.

## 2 Matching Autonomous Dimensions

In this section we illustrate the basic issues of dimension matching and provide a fundamental technique, the d-chase, for the management of matchings.

### 2.1 The Framework of Reference

We refer to a very general data model for multidimensional databases based on the basic notions of *dimension* and *data mart*. A dimension represents a perspective under which data analysis can be performed and consists of entities



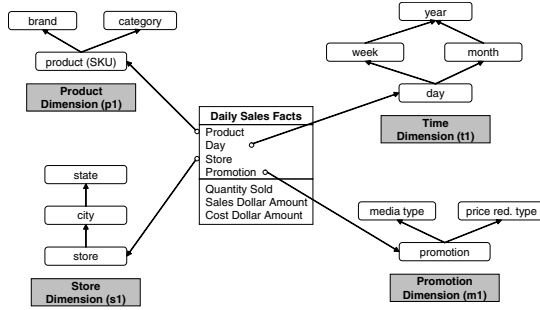


Fig. 1. Sales data mart

called *members*. Members of a dimension can be the days in a time interval or the products sold by a company. Each dimension is organized into a hierarchy  $\preceq$  of *levels*, corresponding to domains grouping dimension members at different granularity. Levels in a product dimension can be the category and the brand of the items sold. The members of the bottom element of a dimension (with respect to  $\preceq$ ) represent real world entities that are called *ground*. Within a dimension, members at different levels are related through a family of *roll-up functions* that map members having a finer grain (e.g., a product) to members having a coarser grain (e.g., a brand) according to  $\preceq$ . A *data mart* associates *measures* to members of dimensions and is used to represent factual data. As an example, Figure 1 shows a *Sales* data mart that has the quantity, the income and the cost of a sale as measures and is organized along the dimensions *Product*, *Time*, *Store*, and *Promotion*.

### 2.2 Properties of Dimension Matchings

The basic problem of the integration of two autonomous data marts is the definition of a *matching* between their dimensions, that is, a (one-to-one) injective partial mapping between the corresponding levels. An example is illustrated in Figure 2, which shows a matching between two heterogeneous geographical dimensions.

We have identified a number of desirable properties that a matching  $\mu$  between two dimensions  $d_1$  and  $d_2$  should satisfy.

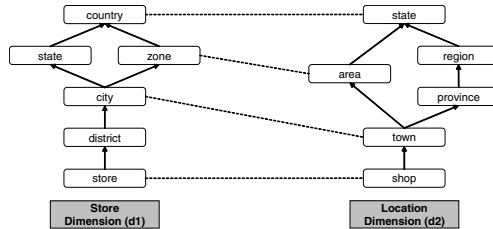


Fig. 2. A matching between two dimensions

- Coherence:  $\mu$  is *coherent* if, for each pair of levels  $l, l'$  of  $d_1$  on which  $\mu$  is defined,  $l \preceq_1 l'$  if and only if  $\mu(l) \preceq_2 \mu(l')$ ;
- Soundness:  $\mu$  is *sound* if, for each level  $l$  of  $d_1$  on which  $\mu$  is defined, there exists a bijection between the members of  $l$  and  $\mu(l)$ ;
- Consistency:  $\mu$  is *consistent* if, for each pair of levels  $l \preceq_1 l'$  of  $d_1$  on which  $\mu$  is defined, the roll-up function from  $l$  to  $l'$  coincides with the roll-up function from  $\mu(l)$  to  $\mu(l')$ .

A total matching that is coherent, sound and consistent is called a *perfect* matching. Clearly, a perfect matching is very difficult to achieve in practice. In many cases however, autonomous dimensions actually share some information. The goal of the tool we have developed is the identification of this common information to perform drill-across operations between heterogeneous data marts.

### 2.3 Chase of Dimensions

The *d-chase* is a powerful technique inspired by an analogous procedure used for reasoning about dependencies in the relational model [2], which can be used to test for consistency and to combine the content of heterogeneous dimensions. Given a matching  $\mu$  between two dimensions  $d_1$  and  $d_2$ , this procedure takes as input a special *matching* tableau  $T_\mu[d_1, d_2]$ , built over the members of  $d_1$  and  $d_2$ , and generates another tableau that, if possible, satisfies the roll-up functions defined for  $d_1$  and  $d_2$ .

A matching tableau  $T_\mu[d_1, d_2]$  has a tuple for each ground member  $m$  of  $d_1$  and  $d_2$  and includes members associated with  $m$  by roll-up functions and possibly *variables* denoting missing information. An example of a matching tableau for the matching between dimensions in Figure 2 is the following.

store	district	city	prov.	region	zone	state	country
1st	$v_1$	NewYork	$v_2$	$v_3$	$v_4$	NY	USA
2nd	Melrose	LosAng.	$v_5$	$v_6$	U2	CA	USA
1er	Marais	Paris	$v_7$	$v_8$	E1	$v_9$	France
1mo	$v_{10}$	Rome	RM	Lazio	E1	$v_{11}$	Italy
1st	$v_{12}$	NewYork	$v_{13}$	$v_{14}$	U1	$v_{15}$	USA
1er	$v_{16}$	Paris	75	IledeFr	E1	$v_{17}$	France

In this example, the first three tuples represent members of  $d_1$  and the others members of  $d_2$ . Note that a variable occurring in a tableau may represents an unknown value (for instance, in the third row, the region in which the store *1er* is located, an information not available in the instance of  $d_1$ ) or an inapplicable value (for instance, in the last row, the district in which the store *1er* is located, a level not present in the scheme of  $d_2$ ). The value of a tuple  $t$  over a level  $l$  will be denoted by  $t[l]$ .

The d-chase modifies values in a matching tableau, by applying *chase steps*. A chase step applies when there are two tuples  $t_1$  and  $t_2$  in  $T$  such that  $t_1[l] = t_2[l]$  and  $t_1[l'] \neq t_2[l']$  for some roll up function from  $l$  to  $l'$  and modifies the  $l'$ -values of  $t_1$  and  $t_2$  as follows: if one of them is a constant and the other is a variable

then the variable is changed (is *promoted*) to the constant, otherwise the values are equated. If a chase step tries to identify two constants, then we say that the d-chase encounters a *contradiction* and the process stops.

By applying the d-chase procedure to the matching tableau above we do not encounter contradictions and obtain the following tableau in which, for instance,  $v_4$  has been promoted to U1 and  $v_{16}$  to Marais.

store	district	city	prov.	region	zone	state	country
1st	$v_1$	NewYork	$v_2$	$v_3$	U1	NY	USA
2nd	Melrose	LosAng.	$v_5$	$v_6$	U2	CA	USA
1er	Marais	Paris	75	IledeFr	E1	$v_9$	France
1mo	$v_{10}$	Rome	RM	Lazio	E1	$v_{11}$	Italy

The d-chase provides an effective way to test for consistency since it is possible to show a matching  $\mu$  between two dimensions  $d_1$  and  $d_2$  is consistent if and only if the chase  $T_\mu[d_1, d_2]$  terminates without encountering contradictions. Moreover, it turns out that if we apply the d-chase procedure over a matching tableau that involves a dimension  $d$  and then project the result over the levels of  $d$ , we obtain the original instance and, possibly, some additional information that has been identified in the other dimension.

### 3 Integration Techniques

In this section we illustrate two different approaches to the problem of the integration of autonomous data marts.

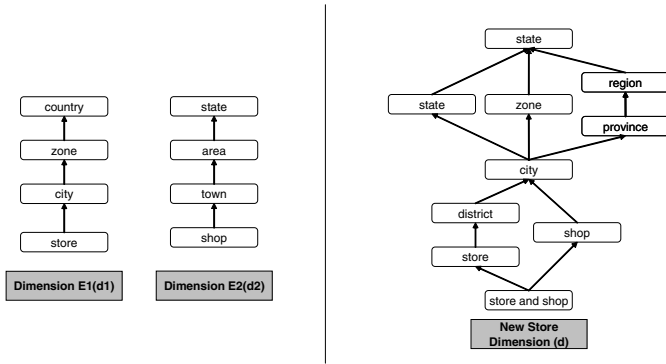
#### 3.1 A Loosely Coupled Approach

In a *loosely coupled integration* scenario, we need to identify the common information between various data sources and perform drill-across queries over the original sources. Therefore, our goal is just to select data that is shared between the sources. Thus, given a pair of dimensions  $d_1$  and  $d_2$  and a matching  $\mu$  between them, the approach aims at deriving two expressions that makes  $\mu$  perfect.

We have elaborated an algorithm that generates two expressions of *dimension algebra* that describe, in an abstract way, data manipulations over dimensions [4]. This algorithm is based on three main steps: (i) a test for coherence that takes advantage of the transitivity of  $\preceq$ , (ii) a test for consistency based on the application of the d-chase, and (iii) the derivation of the selections, projections and aggregations to be performed on the input dimensions in order to select common information.

As an example, the application of this algorithm to the dimension matching reported in Figure 2 returns a pair of expressions that, applied to the original dimensions, generates the dimensions reported on the left hand side of Figure 3.

We have proved that the execution of this algorithm always returns two expressions that correctly compute the intersection of two dimensions if and only if the dimensions are compatible [4].



**Fig. 3.** The dimensions generated by the first algorithm (left) and by the second algorithm (right) on the matching in Figure 2

### 3.2 A Tightly Coupled Approach

In a *tightly coupled integration*, we want to build a materialized view that combines different data sources and perform queries over this view. In this case, given a pair of dimensions  $d_1$  and  $d_2$  and a matching  $\mu$  between them, the integration technique aims at deriving a new dimension obtained by merging the levels involved in  $\mu$  and including, but taking apart, all the other levels.

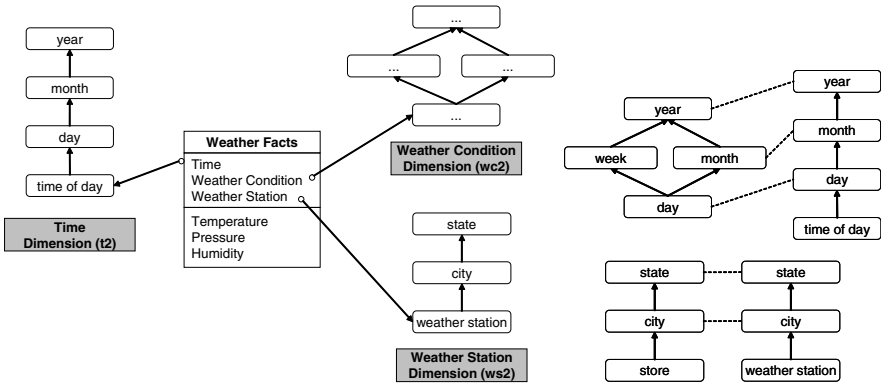
We have elaborated an algorithm that performs this task [4]. This algorithm is also based on three main steps: (i) a test for coherence that takes advantage of the transitivity of  $\preceq$ , (ii) a test for consistency based on the application of the d-chase, and (iii) the derivation of a new dimension obtained by projecting the result of the d-chase over the “union” of the schemes of the input dimensions. If the union of the schemes produces two minimal levels, the algorithm is more involved since it generates an auxiliary bottom level.

As an example, consider again the matching between dimensions in Figure 2 but assume that the level store does not map to the level shop. This means that the corresponding concepts are not related. It follows that the union of the schemes of the two dimensions produces two minimal levels. In this case, the application of algorithm to this matching introduces a new bottom level below store and shop. The scheme of the dimension generated by the algorithm is reported on the right hand side of Figure 3.

We have proved that the execution of this algorithm always returns a dimension  $d$  that “embeds” the original dimensions, in the sense that they can be obtained by applying a dimension expression over  $d$  [4].

### 3.3 Data Mart Integration

*Drill-across* queries are usually used to combine and correlate data from multiple data marts [7]. These queries are based on joining different data marts over common dimensions and so the existence of shared information between data marts is needed in order to obtain meaningful results.

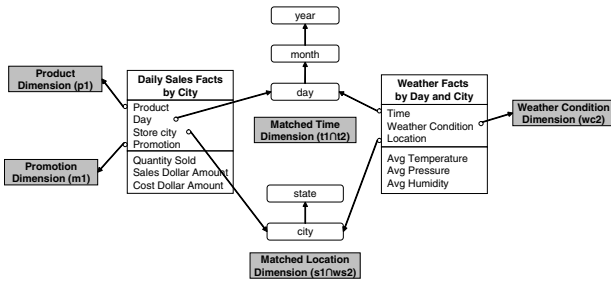


**Fig. 4.** A weather data mart and a matching between its dimensions and the dimensions of the data mart in Figure 4

The loosely coupled approach supports drill-across queries between data marts in that it aims at identifying the intersection between their dimensions. Assume, for instance, that we wish to correlate the Sales data mart reported in Figure 1 with the data mart storing weather information reported in Figure 4, according to the matchings between the time and the location dimensions as indicated on the right hand side of Figure 4.

The application of algorithm illustrated in the previous section to this input checks for compatibility of dimensions and returns two pairs of expressions that select the members in common in the matched dimensions. It turns out that we can join the two data marts to extract daily and city-based data, but hourly or store-based data can not be computed. Moreover, if we apply these expressions to the underlying dimensions before executing the drill-across operation we prevent inconsistencies in subsequent aggregations over the result of the join. It follows that drill-across queries can be defined over the virtual view shown in Figure 5.

The tightly coupled approach aims at combining data from different dimensions by computing their union rather than their intersection. Consider again the example above. If we apply the corresponding algorithm over the same input we obtain two new dimensions that can be materialized and used for both data



**Fig. 5.** A virtual view over the Sales and the Weather data marts

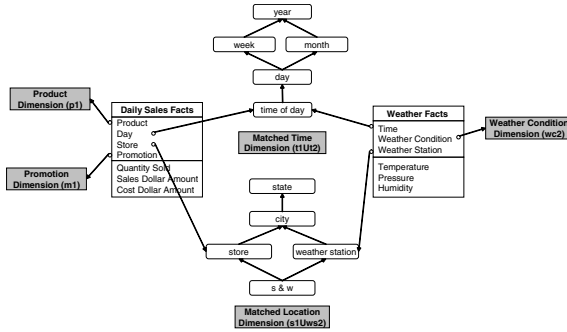


Fig. 6. A materialized view over the merged dimensions

mart. Hence, we can then refer to the homogenous scheme reported in Figure 6 to perform drill-across queries.

#### 4 The Integration Tool

The various techniques described in the previous section have been implemented in an interactive tool (screenshots are reported in Figure 7).

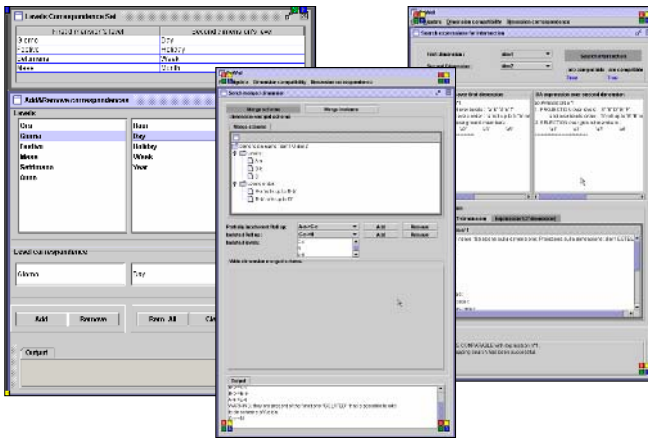


Fig. 7. The integration tool

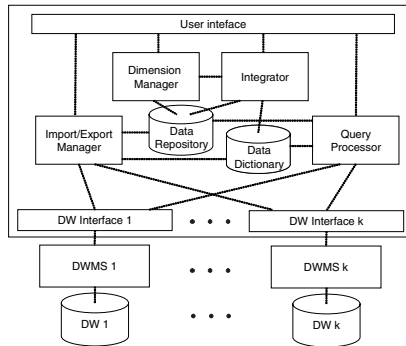
The tool allows the user to:

1. access to data marts stored in a variety of systems (DB2, Oracle, SQL Server, among others);
2. import from these systems metadata describing cubes and dimensions and translate these descriptions into a uniform internal format;

3. specify matchings between heterogeneous dimensions, by means of a graphical interface;
4. suggest possible matching between levels of heterogeneous dimensions;
5. test for coherence, consistency, and soundness of matchings;
6. generate the intersection of two dimensions, according to the the loosely integration approach;
7. merge two dimensions, according to the tightly integrated approach;
8. perform drill-across queries over heterogeneous data marts whose dimensions have been matched according to either the tightly coupled approach or the loosely coupled one.

Function 4 relies on a number of heuristics that try to infer whether two levels of different dimensions can refer to the same concept. Currently, we have followed a rather simple approach based on the name of the levels and on the existence of shared members. We are currently investigating more involved techniques based on the use of data dictionaries. This is however outside the original goal of our project.

The basic components of the tool architecture are reported in Figure 8.



**Fig. 8.** The architecture of the integration tool

A number of external data warehouses stored in different systems are accessed by the tool through *DW Interfaces* that are able to: (i) extract meta data describing the sources, (ii) translate these descriptions into an internal representation that is based on the multidimensional model described in Section 2, and (iii) store this representation in a data dictionary. The *Dimension Manager* is in charge to specify and verify matching between dimensions. The *Integrator* module performs the actual integrations of pair of dimensions according to the either the loosely coupled approach or the tightly coupled one. In the latter case, a new dimension is built and the corresponding members are stored in a local data repository. Finally, the *Query Processor* receives requests of drill-across queries over autonomous data marts and, on the basis of the information available in the internal repositories, performs queries to the external systems through the corresponding DW interfaces.

## 5 Conclusion

We have illustrated in this paper the development of a tool for the integration of heterogeneous multidimensional databases. We have first addressed the problem from a conceptual point of view, by introducing the desirable properties of coherence, soundness and consistency that “good” matchings between dimensions should enjoy. We have then presented two practical approaches to the problem that refer to the different scenarios of loosely and tightly coupled integration. We have then presented a practical tool that implements the various techniques and can be effectively used to perform drill-across queries between heterogeneous data warehouses. The first experimentations demonstrate the effectiveness of the approach and show a reasonable efficiency. We are currently working to further improve the performance of the tool.

We believe that the techniques presented in this paper can be generalized to much more general contexts in which, similarly to the scenario of this study, we need to integrate heterogeneous sources and we possess a taxonomy of concepts that describe their content (e.g., an *ontology*). This is subject of current investigation.

## References

1. A. Abelló, J. Samos, and F. Saltor. On relationships Offering New Drill-across Possibilities. In *ACM Fifth Int. Workshop on Data Warehousing and OLAP (DOLAP 2002)*, pages 7–13, 2002.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. L. Cabibbo and R. Torlone. A logical Approach to Multidimensional Databases. In *Sixth Int. Conference on Extending Database Technology (EDBT’98)*, Springer-Verlag, pages 183–197, 1998.
4. L. Cabibbo and R. Torlone. Integrating Heterogeneous Multidimensional Databases. In *17th Int. Conference on Scientific and Statistical Database Management (SSDBM’05)*, 2005.
5. A. Elmagarmid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann, 1999.
6. M.R. Jensen, T.H. Møller, and T.B. Pedersen. Specifying OLAP Cubes on XML Data. *J. Intell. Inf. Syst.*, 17(2-3): 255–280, 2001.
7. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Second edition, 2002.
8. M. Lenzerini. Data Integration: A Theoretical Perspective. In *21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 233–246, 2002.
9. R.J. Miller, M.A. Hernández, L.M. Haas, L. Yan, C.T.H. Ho, R. Fagin, and L. Popa. The Clio Project: Managing Heterogeneity. *SIGMOD Record*, 30(1): 78–83, 2001.
10. X. Yin and T.B. Pedersen. Evaluating XML-extended OLAP queries based on a physical algebra. In *ACM Int. Workshop on Data Warehousing and OLAP (DOLAP’04)*, pages 73–82, 2004
11. T.B. Pedersen, A. Shoshani, J. Gu, and C.S. Jensen. Extending OLAP Querying to External Object Databases. In *Int. Conference on Information and Knowledge Management*, pages 405–413, 2000.
12. E. Rahm and P.A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.



# An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse

Ladjel Bellatreche<sup>1</sup> and Kamel Boukhalfa<sup>2</sup>

<sup>1</sup> LISI/ENSMA - Poitiers University - France  
bellatre@ensma.fr

<sup>2</sup> Université de Laghouat - Algeria  
k.boukhalfa@mail.lagh-univ.dz

**Abstract.** The problem of selecting an optimal fragmentation schema of a data warehouse is more challenging compared to that in relational and object databases. This challenge is due to the several choices of partitioning star or snowflake schemas. Data partitioning is beneficial if and only if the fact table is fragmented based on the partitioning schemas of dimension tables. This may increase the number of fragments of the fact tables dramatically and makes their maintenance very costly. Therefore, the right selection of fragmenting schemas is important for better performance of OLAP queries. In this paper, we present a genetic algorithm for schema partitioning selection problem. The proposed algorithm gives better solutions since the search space is constrained by the schema partitioning. We conduct several experimental studies using the APB-1 release II benchmark for validating the proposed algorithm.

## 1 Introduction

The main characteristics of data warehouses are (1) their data complexity due to the presence of hierarchies between attributes, (2) the huge amount of data, and (3) the complexity of their queries due to the presence of join and aggregate operations. Several queries optimization techniques were proposed in the literature and supported by commercial systems. These techniques can be classified into two categories: (1) *redundant-structures* and (2) *non redundant-structures*. Techniques in the first category compete for the same resource representing the storage cost and incur maintenance overhead in the presence of updates [12]. We can cite: materialized views and indexes. Techniques in the second category do not require an extra space compare to those in the first category. We can cite vertical and horizontal partitioning [11]. Horizontal partitioning (HP) allows access methods such as tables, indexes and materialized views to be partitioned into disjoint sets of rows that are stored and accessed separately. On the other hand, vertical partitioning allows a table to be partitioned into disjoint sets of columns. Like indexes and materialized views, both kinds of partitioning can significantly impact the performance of the workload i.e., queries and updates that execute against the database system, by reducing cost of accessing and processing data. In this paper, we are interesting to a non redundant structure,

which is the HP. Several work and commercial systems show its utility and impact in optimizing OLAP queries [11,3,2,8,13]. But none study has formalized the problem of selecting a HP schema to speed up a set of queries and proposed selection algorithms. In this paper, we use fragmentation and partitioning interchangeably. HP in relational data warehouses is more challenging compared to that in relational and object databases. This challenge is due to the several choices of partitioning schemas <sup>1</sup> that can be found:

1. *partition only the dimension tables* using simple predicates defined on these tables <sup>2</sup>. This scenario is not suitable for OLAP queries, because the sizes of dimension tables are generally small compare to the fact table. Therefore, any partitioning that does not take into account the fact table is discarded.
2. *partition only the fact table* using simples predicates defined on this table. Note that a fact relation stores foreign keys and raw data which is usually never contain descriptive (textual) attributes because it is designed to perform arithmetic operations. On the other hand, in a relational data warehouse, most of OLAP queries access dimension tables first and then the fact table. This choice is also discarded.
3. *partition some/all dimension tables using their predicates, and then partition the fact table* based on the fragmentation schemas of dimension tables. This approach is best in applying partitioning in data warehouses. Because it takes into consideration star join queries requirements (these queries impose restrictions on the dimension values that are used for selecting specific facts; these facts are further grouped and aggregated according to the user demands. The major bottleneck in evaluating such queries has been the join of a large fact table with the surrounding dimension tables [13]). In our study, we opt for last solution.

To show the procedure to fragment a fact table using this scenario, suppose that a dimension table  $D_i$  is fragmented into  $m_i$  fragments:  $\{D_{i1}, D_{i2}, \dots, D_{im_i}\}$ , where each fragment  $D_{ij}$  is defined as:  $D_{ij} = \sigma_{cl_j^i}(D_i)$ , where  $cl_j^i$  ( $1 \leq i \leq g, 1 \leq j \leq m_i$ ) represents a conjunction of simple predicates. Thus, the fragmentation schema of the fact table  $F$  is defined as follows:  $F_i = F \times D_{1i} \times D_{2i} \times \dots \times D_{gi}$ , with  $\times$  represents the semi join operation. In order to illustrate this procedure, let consider a star schema with three dimension table (Customer, Time and Product) and one fact table Sales. Suppose that the dimension table Customer is fragmented into two fragments Cust\_1 and Cust\_2 defined by the following clauses:  $Cust\_1 = \sigma_{Sex=M'}(Customer)$  and  $Cust\_2 = \sigma_{Sex=F'}(Customer)$ . Therefore the fact table Sales can be fragmented using the fragmentation schema of the dimension table Customer into two fragments Sales\_1 and Sales\_2 such as:  $Sales\_1 = Sales \times Cust\_1$  and  $Sales\_2 = Sales \times Cust\_2$ .

The initial star schema (Sales, Customer, Product, Time) is represented as the juxtaposition of two sub star schemas  $S_1$  et  $S_2$  such as:  $S_1 : (Sales\_1, Cust\_1,$

<sup>1</sup> The fragmentation schema is the result of the data partitioning process.

<sup>2</sup> A simple predicate  $p$  is defined by:  $p : A_i \theta Value$ , where  $A_i$  is an attribute,  $\theta \in \{=, <, >, \leq, \geq\}$ , and  $value \in Dom(A_i)$

Product, Time) (sales activities for only male customers) et  $S_2$  : (Sales\_2, Cust\_2, Product, Time) (sales activities for only female customers).

To the best of our knowledge, the proposed work is the first article that addresses horizontal fragmentation schema selection problem in relational data warehouses and uses a genetic algorithm to select a right solution that minimizes the performance of OLAP queries, and reduces the maintenance cost.

This paper is divided in five sections: The section 2 formalizes the fragmentation selection problem in data warehouses modeled using star schemas. Section 3 presents a genetic algorithm with its four steps (selection, coding, mutation, and fitness function). Section 4 gives the experimental results using benchmark APB-1 release II benchmark. The Section 5 concludes the paper by summarizing the mains results and suggesting future work.

## 2 Complexity of Generated Fragments of the Fact Table

Let a star schema with  $d$  dimension tables and a fact table. Let  $g$  ( $g \leq d$ ) be the number of fragmented dimension tables. The number of horizontal fragments of the fact table (denoted by  $N$ ) is given by:  $\mathbf{N} = \prod_{i=1}^g \mathbf{m}_i$ , where  $m_i$  is the number of fragments of dimension table  $D_i$ . This fragmentation technique generates a large number of fragments of the fact table. For example, suppose we have: *Customer* dimension table partitioned into 50 fragments using the State attribute<sup>3</sup>, *Time* into 36 fragments using the Month attribute, and *Product* into 80 fragments using Package\_type attribute, therefore the fact table will be fragmented into 144 000 fragments ( $50 \times 36 \times 80$ ). Consequently, instead of managing one star schema, we will manage 144 000 sub star schemas. It will be very hard for the data warehouse administrator (DWA) to maintain all these sub-star schemas.

Therefore it is necessary to reduce the number of fragments of the fact table in order to guarantee two main objectives: (1) avoid an explosion of the number of the fact fragments and (2) ensure a good performance of OLAP queries. To satisfy the first objective, we give to DWA the possibility to choose the number of fragment maximal that he/she can maintain (threshold  $W$ ). For the second one, we can increase the number of fragment so that the global performance will be satisfied. The problem of selecting an optimal fragmentation schema consists in finding a compromise between the maintenance cost and the performance cost.

In order to satisfy this compromise, we use genetic algorithms [1,4] since they explore a large search space. Our problem is similar to the problems multiprocessor document allocation [6], and data replication [9], where genetic algorithms gave good results.

## 3 Genetic Algorithms

Genetic algorithms (GAs) [7], are search methods based on the evolutionary concept of natural mutation and the survival of the fittest individuals. Given a

<sup>3</sup> case of 50 states in the U.S.A.

well-defined search space they apply three different genetic search operations, namely, *selection*, *crossover*, and *mutation*, to transform an initial population of chromosomes, with the objective to improve their quality. Fundamental to the GA structure is the notion of chromosome, which is an *encoded* representation of a feasible solution, most commonly a bit string. Before the search process starts, a set of chromosomes is initialized to form the first generation. Then the three genetic search operations are repeatedly applied, in order to obtain a population with better characteristics. An outline of a generic GA is as follows:

```

Generate initial population ;
Perform selection step;
while stopping criterion not met do
    Perform crossover step;
    Perform mutation step;
    Perform selection step ;
end while.

```

Report the best chromosome as the final solution. We demonstrate the design of our algorithm in details by presenting our encoding mechanism and then the selection, crossover and mutation operators.

### 3.1 Representation of Solutions

Representation of solution or chromosome is one of the key issues in problem solving. In our study, a solution represents a fragmentation schema. Note that any fragmentation algorithm needs application information defined on the tables that have to be partitioned. The information is divided into two categories [10]: quantitative and qualitative. Quantitative information gives the selectivity factors of selection predicates and the frequencies of queries accessing these tables ( $Q = \{Q_1, \dots, Q_n\}$ ). Qualitative information gives the selection predicates defined on dimension tables. Before representing each solution, the following tasks should be done:

1. extraction of all simple predicates used by the  $n$  queries,
2. assignment to each dimension table  $D_i (1 \leq i \leq d)$ , its set of simple predicates ( $SSPD_i$ ),
3. each dimension table  $D_i$  having  $SSPD_i = \phi$  cannot be fragmented. Let  $D_{candidate}$  be the set of all dimension tables having a non-empty  $SSPD_i$ . Let  $g$  be the cardinality of  $D_{candidate}$  ( $g \leq d$ ),
4. use the COM\_MIN algorithm [10] to each dimension table  $D_i$  of  $D_{candidate}$ . This algorithm takes a set of simple predicates and then generates a set of complete and minimal.

**Representation of Horizontal Fragments** Note each fragmentation predicate has a domain values. The clauses of simple predicates representing horizontal fragments defines partitions of each attribute domain into sub domains. The cross product of partitions of an attribute by all predicates determines a partitioning of the domains of all the attributes into sub domains.

*Example 1.* Consider three fragmentation attributes <sup>4</sup> Age, Gender and City of dimension table Customer and one attribute Season of dimension table Time. The domains of these attributes are defined as follows:  $Dom(Age) = ]0, 120]$ ,  $Dom(Season) = \{“Summer”, “Spring”, “Autumn”, “Winter”\}$ , and  $Dom(Gender) = \{‘M’, ‘F’\}$ . Suppose that on attribute Age, three simple predicates are defined as follows:  $p_1 : Age \leq 18$ ,  $p_2 : Age \geq 60$ , and  $p_3 : 18 < Age < 60$ . The domain of this attribute ( $]0, 120]$ ) is then partitioned into three sub domains ( $p_1$ ,  $p_2$ , and  $p_3$ ).  $Dom(Age) = d_{11} \cup d_{12} \cup d_{13}$ , with  $d_{11} = ]0, 18]$ ,  $d_{12} = ]18, 60]$ ,  $d_{13} = [60, 120]$ . Similarly, the domain of Gender attribute is decomposed into two sub domains:  $Dom(Gender) = d_{21} \cup d_{22}$ , with  $d_{21} = \{‘M’\}$ ,  $d_{22} = \{‘F’\}$ . Finally, domain of Season is partitioned into four sub domains :  $Dom(Season) = d_{31} \cup d_{32} \cup d_{33} \cup d_{34}$ , where  $d_{31} = \{“Summer”\}$ ,  $d_{32} = \{“Spring”\}$ ,  $d_{33} = \{“Autumn”\}$ , and  $d_{34} = \{“Winter”\}$ .

Each fragmentation attribute can be represented by an array with  $n$  cells, where  $n$  corresponds to number of its sub domains. The values of these cells are between 1 and  $n$ . If two cells have the same values, then they will be merged to form only one. Each fragmentation schema is represented by a multi-dimensional arrays. Suppose we have the following representation Gender: (1, 1), Season(2, 1, 3, 3) and Age (2, 1, 2). We can deduce that the fragmentation of the data warehouse is not performed using the attribute Gender, because all its sub domains have the same value. Consequently, the warehouse will be fragmented using only Season and Age. For Season attribute, three simple predicates are possible:  $P_1 : Season = “Spring”$ ,  $P_2 : Season = “Summer”$ , and  $P_3 : Season = “autumn” \vee Season = “Winter”$ . For Age attribute, two predicates are possible:  $P_4 : Age \leq 18 \vee Age \geq 60$  et  $P_5 : 18 < Age < 60$  With these simple predicates, the data warehouse can be fragmented into six fragments defined by the following clauses:  $Cl_1 : P_1 \wedge P_4$ ;  $Cl_2 : P_1 \wedge P_5$ ;  $Cl_3 : P_2 \wedge P_4$ ;  $Cl_4 : P_2 \wedge P_5$ ;  $Cl_5 : P_3 \wedge P_4$ ; and  $Cl_6 : P_3 \wedge P_5$ . The coding that we proposed satisfies the correctness rules (completeness, reconstruction and disjointness [10]) and the new chromosomes generated by cross over operations belong to the relevant sub domains (it does not generate invalid solutions). This coding can be used to represent fragments of dimension tables and fact table.

### 3.2 Selection Mechanism

Selection in genetic algorithms determines the probability of individuals being selected for reproduction. The principle here is to assign higher probabilities to filter individuals. The roulette wheel method is used in our algorithm (it allocate a sector of the wheel equaling to the  $i^{th}$  chromosome and creating an offspring if a generated number in the range of 0 to falls inside the assigned sector of the string). In this method, each chromosome is associated with its fitness value calculated using the cost model defined in section 3.4. The chromosomes with high fitness values have chances to be selected.

<sup>4</sup> A fragmentation attribute is an attribute participating in the fragmentation process

### 3.3 Types of Crossover

We selected a two-point crossover mechanism to include in our GA for the following reason: note that fragments are represented by arrays. The chromosomes are crossed over once for each predicate. If the crossover is done over one chromosome, the predicates with high number (example of city) will have a probability greater than predicate with small predicate like gender. This operation is applied till none reduction of the number of suitable fragments of fact table ( $W$ ). The rationale behind crossover operation, is that after the exchange of genetic materials, it is very likely that the two newly generated chromosomes will possess the good characteristics of both their parents (building-block hypothesis [7]).

### 3.4 Fitness Value

The quality of each chromosome is measured by computing its fitness value. This function gives a percentage for each performance parameters (respect of threshold and performance of queries). A number of points is assigned to these two parameters: (1) *threshold*: 55 points over 100 are assigned (by default). If the number of obtained fragments is equal plus or minus 5 per cent of the threshold, then all points will be assigned. Otherwise, less points will be assigned to this parameter, and (2) *performance of queries*: a number of points (45) is assigned to all queries in an uniform manner (we have used 15 queries). To compute the cost of each query, we developed a cost model calculating the number of inputs and outputs. As in the previous case, we assign all points (3 per query) if the cost of a query is less than a given number. If the number of IOs increases, less we assign points, following a linearly decreasing function. When the number of IOs of a given query is very high, none point is assigned.

To estimate the cost of queries, we assume that all dimension tables are in the main memory. Let  $D^{sel} = \{D_1^{sel}, \dots, D_k^{sel}\}$  be the set of dimension tables having selection predicates, where each selection predicate  $p_j$  (defined on a dimension table  $D_i$ ) has a selectivity factor denoted by  $Sel_{D_i}^{p_j}$  ( $Sel_{D_i}^{p_j} \in [0, 1]$ ). For each predicate  $p_j$ , we define its selectivity factor on the fact table, denoted by  $Sel_F^{p_j}$  ( $Sel_{D_i}^{p_j} \neq Sel_F^{p_j}$ ). For example, if we consider the selection predicate Gender="Female" defined on the dimension table. Suppose that its selectivity factor is 0.4. This means that 40% of salespersons are female and 60% are male. But, female sales activities may represent 70% of the whole sales. To execute a query  $Q$  over a partitioned star schema  $\{S_1, S_2, \dots, S_N\}$ , we shall identify the relevant sub star schema(s). To do so, we introduce a boolean variable denoted by  $valid(Q, S_i)$  and defined as follows:  $valid(Q, S_i) = 1$  if the sub star schema  $S_i$  is relevant for  $Q$ , 0 otherwise. The number of IOs for executing a query  $Q$  over a partitioned star schema is given by:  $Cost(Q) = \sum_{j=1}^N valid(Q, S_j) \prod_{i=1}^{M_j} \left( \frac{Sel_F^{p_i} \times ||F|| \times L}{PS} \right)$ , where,  $M_j$ ,  $F$ ,  $L$  and  $PS$  represent the number of selection predicates defining the fact fragment of the sub star schema  $SDE_j$ , the number of tuples present in a fact table  $F$ , the width, in bytes, of a tuple of a table  $F$  and the page size of the file system (in bytes), respectively. In this study, the selectivity factors are chosen using an uniform distribution (UD) and a non uniform distribution (NUD).

### 3.5 The Mutation

Although crossover can put good genes together to generate better offspring, it cannot generate new genes. Mutation is needed to create new genes that may not be present in any member of a population and enables the algorithm to reach all possible solutions (in theory) in the search space. Mutation is an operation aiming at restoring lost genetic material and is performed in our algorithm by simply flipping every bit with a certain probability, called the mutation rate. We have chosen a mutation rate between 30 and 6 percent (rate often used). Mutations are done for fragmentation attributes. Initialization of the first generation is performed by randomly generating half of the population while the rest is obtained from the solutions previously found by algorithm. In practice, there could be more intervals distinct or a merged intervals. In the same way, mutations could occur on several attributes of the individual.

## 4 Experimental Studies

In our experiments, we use the dataset from the APB1 benchmark [5]. The star schema of this benchmark has one fact table *Actvars* ( $||Actvars|| = 24786000$  tuples, with a width = 74) and four dimension tables: *Prodlevel* ( $||Prodlevel|| = 9\ 000$  tuples, with a width = 72), *Custlevel* ( $||Custlevel|| = 900$  tuples, with a width = 24), *Timelevel* ( $||Timelevel|| = 24$  tuples, with a width = 36), and *Chanlevel* ( $||Chanlevel|| = 9$  tuples, with a width = 24). This warehouse has been populated using the generation module of APB1. Our simulation software was built using Visual C performed under a Pentium IV 2,8 Ghz microcomputer (with a memory of 256 Mo). We have considered 15 queries. Each query has selection predicates, where each one has its selectivity factor. The crossover and mutations rates used in our experiments are 70% and 30% in the beginning. After several generation, the mutation rate of 6% was used to avoid a redundant search. We have used 1 500 generations (40 chromosomes per generation). 9 fragmentation attributes were considered.

If the DWA chooses the threshold as 2000, the dimension tables will be fragmented as follows: table *Prodlevel* in 48 fragments, table *Timelevel* in 7 fragments, table *Custlevel* in 2 fragments, table *Chanlevel* in 3 fragments, and

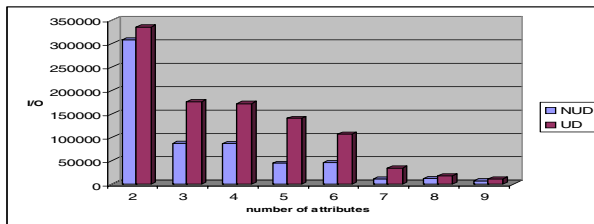
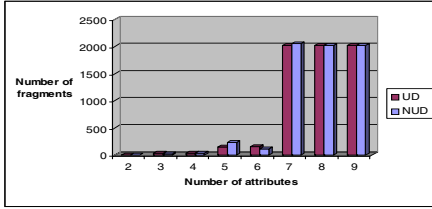
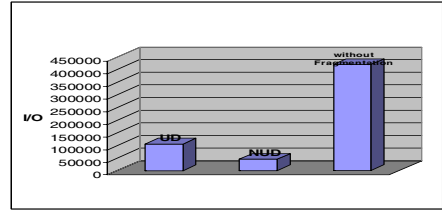


Fig. 1. Impact of fragmentation attributes on the query performance



**Fig. 2.** The impact of the number of fragmentation attributes



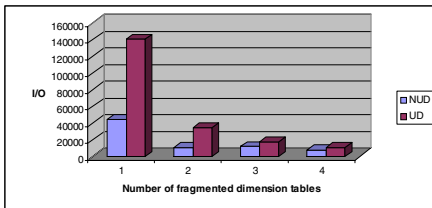
**Fig. 3.** Savings in query processing cost versus type of distribution

the fact table in 2016 fragments. Figure 1 shows the evolution of IOs over the number of fragmentation attributes. The results show the impact of this number on the performance of queries. We note also that the non uniform distribution gives better performance than the uniform distribution.

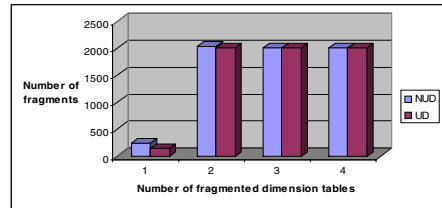
In Figure 2, we have studied the effect of the number of fragmentation attributes over the number of fact table. We realize that the type of distribution does not have an effect on the total number of fragments.

Figure 3 shows the effect of the horizontal fragmentation and its role in reducing the global cost of executing a set of queries. These results confirmed the existing theoretical studies.

In Figure 4, we have studied the effect of the dimension tables participating on the fragmentation process. The performance of OLAP queries is proportional with the number of these tables. Figure 5 shows that the number of fragments of the fact table increases when the the number of dimension tables participating on the fragmentation process increases. But our algorithm controls this augmentation. When we used less than six fragmentation attributes, the rate between the number of fragments return by the algorithm and the possible number of fragment is high (more than 35%) because the possible number of fragments is small. From six attributes, this rate is small (less than de 2%) when we used 9 attributes (Figure 6). To get a better performance of queries, the threshold is varied, and experiments show that this performance is obtained when threshold is equals 4000.

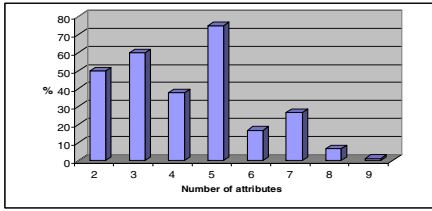


**Fig. 4.** The effect of number of fragmented dimension tables on query processing cost

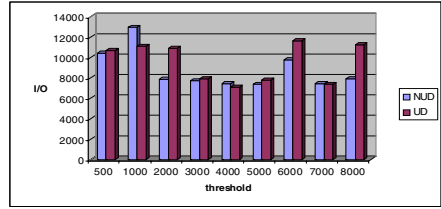


**Fig. 5.** Number of fragments versus number of fragmented dimension tables

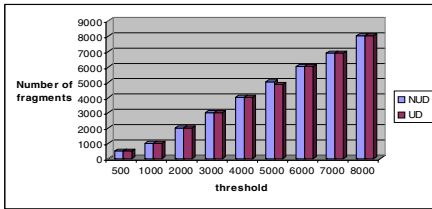




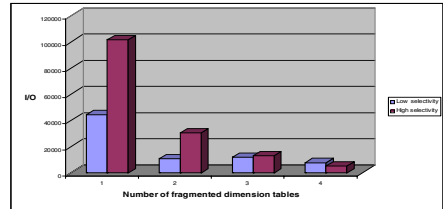
**Fig. 6.** Number of attributes versus number of possible fragments



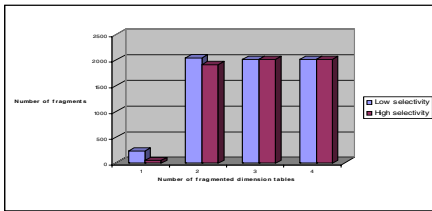
**Fig. 7.** Evolution of IO cost over the threshold



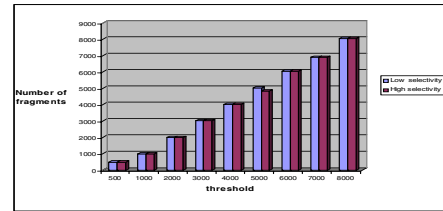
**Fig. 8.** Number of fragments versus the threshold



**Fig. 9.** Number of IOs versus selectivity type and number of fragmented dimension tables



**Fig. 10.** Number of fragments versus the selectivity type and the number of fragmented dimension



**Fig. 11.** Number of fragments versus the selectivity type and the threshold

Note that the number of fragments increases when the threshold increases, but it remains closer to the threshold (Figure 8).

In Figures 9, 10 and 11 we changed the selectivity factors of predicates in order to see their effect on the number of fragments and performance of queries. We realized that when we increase these factors, the number of IOs increases. This is due to the fact that an high selectivity implies a large number of tuples satisfying predicates. But the selectivity factors do not have a strong effect on the final number of fragments.

## 5 Conclusion

In this paper, we have formalized the problem of selecting an horizontal fragmentation schema in relational data warehousing environments. First we have developed a methodology for fragmenting a star schema using the fragmentation schemas of the dimension tables. We have also shown the complexity of the generated fragments of the fact table. This number can be very huge and then it will be difficult for the data warehouse administrator to maintain all fragments. To reduce this number and guarantee a good performance, we proposed a genetic algorithm. Before applying this algorithm, we presented a coding mechanism for all possible solutions. A cost model for evaluating the cost of a set of frequently queries performed on a fragmented star schema is developed. This model is also used to measure the quality of the final solution. Finally, we conducted experiments to show the utility of the horizontal fragmentation and capture different points that can have effect on the performance of OLAP queries and respecting the threshold fixed by the administrator.

It will be interested to develop or adapt our algorithm to take into account the dynamic aspect of a warehouse due to the evolution of the schema and queries.

## References

1. T. Bäck. *Evolutionary algorithms in theory and practice*. Oxford University Press, New York, 1995.
2. L. Bellatreche, K. Karlapalem, and M. Mohania. What can partitioning do for your data warehouses and data marts. *Proceedings of the International Database Engineering and Application Symposium (IDEAS'2000)*, pages 437–445, September 2000.
3. L. Bellatreche, M. Schneider, H. Lorinquer, and M. Mohania. Bringing together partitioning, materialized views and indexes to optimize performance of relational data warehouses. *Proceeding of the International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2004)*, pages 15–25, September 2004.
4. K. P. Bennett, M. C. Ferris, and Y. E. Ioannidis. A genetic algorithm for database query optimization. in *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 400–407, July 1991.
5. OLAP Council. Apb-1 olap benchmark, release ii. <http://www.olapcouncil.org/research/bmarkly.htm>, 1998.
6. O. Frieder and H.T. Siegelmann. Multiprocessor document allocation : A genetic algorithm approach. *IEEE Transactions on Knowledge and Data Engineering*, 9(4):640–642, July 1997.
7. J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.
8. P. Kalnis and D. Papadias. Proxy-server architecture for olap. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001.
9. T. Loukopoulos and I. Ahmad. Static and adaptive distributed data replication using genetic algorithms. in *Journal of Parallel and Distributed Computing*, 64(11):1270–1285, November 2004.

10. M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems : Second Edition*. Prentice Hall, 1999.
11. A. Sanjay, V. R. Narasayya, and B. Yang. Integrating vertical and horizontal partitioning into automated physical database design. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 359–370, June 2004.
12. A. Sanjay, C. Surajit, and V. R. Narasayya. Automated selection of materialized views and indexes in microsoft sql server. *Proceedings of the International Conference on Very Large Databases*, pages 496–505, September 2000.
13. T. Stöhr, H. Märtens, and E. Rahm. Multi-dimensional database allocation for parallel data warehouses. *Proceedings of the International Conference on Very Large Databases*, pages 273–284, 2000.

# Using Schema Transformation Pathways for Incremental View Maintenance

Hao Fan

School of Computer Science & Information Systems, Birkbeck College,  
University of London, Malet Street, London WC1E 7HX  
hao@dcs.bbk.ac.uk

**Abstract.** With the increasing amount and diversity of information available on the Internet, there has been a huge growth in information systems that need to integrate data from distributed, heterogeneous data sources. Incrementally maintaining the integrated data is one of the problems being addressed in data warehousing research. This paper presents an incremental view maintenance approach based on schema transformation pathways. Our approach is not limited to one specific data model or query language, and would be useful in any data transformation/integration framework based on sequences of primitive schema transformations.

## 1 Introduction

Data warehouses collect data from distributed, autonomous and heterogeneous data sources into a central repository to enable analysis and mining of the integrated information. When data sources change, the data warehouse, in particular the materialised views in the data warehouse, must be updated also. This is the problem of view maintenance in data warehouses. In contrast to operational database systems handling day-to-day operations of an organisation and dealing with small changes to the databases, data warehouses support queries by non-technical users based on long-term, statistical information integrated from a variety of data sources, and do not require the most up to date operational version of all the data. Thus, data warehouses are normally refreshed periodically and updates to the primary data sources do not have to be propagated to the data warehouse immediately.

AutoMed<sup>1</sup> is a heterogeneous data transformation and integration system which offers the capability to handle data integration across multiple data models. In the AutoMed approach, the integration of schemas is specified as a sequence of primitive schema transformation steps, which incrementally add, delete or rename schema constructs, thereby transforming each source schema into the target schema. We term the sequence of primitive transformations steps defined for transforming a schema  $S_1$  into a schema  $S_2$  a *transformation pathway* from  $S_1$  to  $S_2$ .

In previous work (see [7]), we discussed how AutoMed metadata can be used to express the schemas and the cleansing, transformation and integration processes in heterogeneous data warehousing environments. In this paper, we will describe how AutoMed metadata can be used for maintaining the warehouse data.

---

<sup>1</sup> See <http://www.doc.ic.ac.uk/automed/>

Materialised warehouse views need to be maintained either when the data of a data source changes, or if there is an evolution of a data source schema. In previous work (see [8]), we showed that AutoMed transformation pathways can be used to handle schema evolutions in a data warehouse. In this paper, we will focus on refreshing materialised warehouse views at the data level.

Materialised views can be refreshed by recomputing from scratch or, on the other hand, by only computing the changes to the views rather than all the view data, which is termed *incremental view maintenance* (IVM). Incrementally refreshing a view can be significantly cheaper than fully recomputing the view, especially if the size of the view is large compared to the size of the change.

The problem of view maintenance at the data level has been widely discussed in the literature. Dong and Gupta *et al* give good surveys of this problem [6,10]. Colby *et al*, Griffin *et al* and Quass present propagation formulae based on relational algebra operations for incrementally maintaining views with duplicates and aggregations [5,9,16]. Zhuge *et al* consider the IVM problem for a single-source data warehouses and defines the ECA algorithm [20]. The IVM approaches for a multi-source data warehouse include the Strobe algorithm [21], and the SWEEP and Nested SWEEP algorithms [1]. The view maintenance approach discussed by Gupta and Quass *et al* is to make views *self-maintainable*, which means that materialised views can be refreshed by only using the content of the views and the updates to the data sources, and not requiring to access the data in any underlying data source [11,17]. Such a view maintenance approach usually needs auxiliary materialised views to store additional information.

Our IVM approach presented in this paper is based on AutoMed schema transformation pathways, which is not limited to one specific data model or query language, and would be useful in any data transformation/integration framework based on sequences of primitive schema transformations.

The outline of this paper is as follows. Section 2 gives an overview of AutoMed, as well as a data integration example. Section 3 presents our IVM formulae and algorithms using AutoMed schema transformations. Section 4 gives our concluding remarks and directions of further work.

## 2 Overview of AutoMed

AutoMed supports a low-level hypergraph-based data model (HDM). Higher-level modelling languages are defined in terms of this HDM. For example, previous work has shown how relational, ER, OO [13], XML [18], flat-file [3] and multidimensional [7] data models can be so defined. An HDM schema consists of a set of nodes, edges and constraints, and each modelling construct of a higher-level modelling language is specified as some combination of HDM nodes, edges and constraints. For any modelling language  $\mathcal{M}$  specified in this way, via the API of AutoMed's Model Definitions Repository [3], AutoMed provides a set of primitive schema transformations that can be applied to schema constructs expressed in  $\mathcal{M}$ . In particular, for every construct of  $\mathcal{M}$  there is an **add** and a **delete** primitive transformation which add to/delete from a schema an instance of that construct. For those constructs of  $\mathcal{M}$  which have textual names, there is also a **rename** primitive transformation.

In AutoMed, schemas are incrementally transformed by applying to them a sequence of primitive transformations  $t_1, \dots, t_r$ . Each primitive transformation adds, deletes or renames just one schema construct, expressed in some modelling language. Thus, the intermediate (and indeed the target) schemas may contain constructs of more than one modelling language.

Each **add** or **delete** transformation is accompanied by a query specifying the extent of the new or deleted construct in terms of the rest of the constructs in the schema. This query is expressed in a functional query language IQL<sup>2</sup>. The queries within **add** and **delete** transformations are used by AutoMed's Global Query Processor to evaluate an IQL query over a global schema in the case of a virtual data integration scenario. In the case that the global schema is materialised, AutoMed's Query Evaluator can be used directly on the materialised data.

## 2.1 Simple IQL

In order to illustrate our IVM algorithm, we use a subset of IQL, *Simple IQL* (SIQL), as the query language in this paper. More complex IQL queries can be encoded as a series of transformations with SIQL queries on intermediate schema constructs. We stress that although illustrated within a particular query language syntax, our IVM algorithms could also be applied to schema transformation pathways involving queries expressed in other query languages supporting operations on set, bag and list collections.

Supposing  $D, D_1, \dots, D_n$  denote bags of the appropriate type (base collections), SIQL supports the following queries: `group D` groups a bag of pairs  $D$  on their first component. `distinct D` removes duplicates from a bag. `f D` applies an aggregation function  $f$  (which may be `max`, `min`, `count`, `sum` or `avg`) to a bag. `gc f D` groups a bag  $D$  of pairs on their first component and applies an aggregation function  $f$  to the second component. `++` is the bag union operator and `--` is the bag *monus* operator [2]. SIQL comprehensions are of three forms:  $[\bar{x}|\bar{x}_1 \leftarrow D_1; \dots; \bar{x}_n \leftarrow D_n; C_1; \dots; C_k]$ ,  $[\bar{x}|\bar{x} \leftarrow D_1; \text{member } D_2 \bar{y}]$ , and  $[\bar{x}|\bar{x} \leftarrow D_1; \text{not}(\text{member } D_2 \bar{y})]$ . Here, each  $\bar{x}_1, \dots, \bar{x}_n$  is either a single variable or a tuple of variables.  $\bar{x}$  is either a single variable or value, or a tuple of variables or values, and must include all of variables appearing in  $\bar{x}_1, \dots, \bar{x}_n$ . Each  $C_1, \dots, C_k$  is a condition not referring to any base collection. Also, each variable appearing in  $\bar{x}$  and  $C_1, \dots, C_k$  must also appear in some  $\bar{x}_i$ , and the variables in  $\bar{y}$  must appear in  $\bar{x}$ . Finally, a query of the form `map ( $\lambda \bar{x}.e$ ) D` applies to each element of a collection  $D$  an anonymous function defined by a lambda abstraction  $\lambda \bar{x}.e$  and returns the resulting collection.

Comprehension syntax can express the common algebraic operations on collection types such as sets, bags and lists [4] and such operations can be readily expressed in SIQL. In particular, let us consider *selection* ( $\sigma$ ), *projection* ( $\pi$ ), *join* ( $\bowtie$ ), and *aggregation* ( $\alpha$ ) (*union* ( $\cup$ ) and *difference* ( $-$ ) are directly supported in SIQL via the `++` and `--` operators). The general form of a Select-Project-Join (SPJ) expression is  $\pi_A(\sigma_C(D_1 \bowtie \dots \bowtie D_n))$  and this can be expressed as follows in comprehension syntax:  $[\bar{A}|\bar{x}_1 \leftarrow D_1; \dots; \bar{x}_n \leftarrow D_n; C]$ . However, since in general the tuple of variables  $A$

<sup>2</sup> IQL is a comprehensions-based functional query language. Such languages subsume query languages such as SQL and OQL in expressiveness [4]. We refer the reader to [12,15] for details of IQL and references to work on comprehension-based functional query languages.

may not contain all the variables appearing in  $\overline{x_1}, \dots, \overline{x_n}$  (as is required in SIQL), we can use the following two transformation steps to express a general SPJ expression in SIQL, where  $\overline{x}$  includes all of the variables appearing in  $\overline{x_1}, \dots, \overline{x_n}$ :

$$\begin{aligned} v1 &= [\overline{x} | \overline{x_1} \leftarrow D_1; \dots; \overline{x_n} \leftarrow D_n; C] \\ v &= \text{map } (\lambda \overline{x}. A) v1 \end{aligned}$$

The algebraic operator  $\alpha$  applies an aggregation function to a collection and this functionality is captured by the  $\text{gc}$  operator in SIQL. E.g., supposing the scheme of a collection  $D$  is  $D(A_1, A_2, A_3)$ , an expression  $\alpha_{A_2, f(A_3)}(D)$  is expressed in SIQL as:

$$\begin{aligned} v1 &= \text{map } (\lambda \{x1, x2, x3\} . \{x2, x3\}) D \\ v &= \text{gc } f v1 \end{aligned}$$

## 2.2 An Example Data Integration

In this paper, we will use schemas expressed in a simple relational data model to illustrate our techniques. However, we stress that these techniques are applicable to schemas defined in *any* data modelling language having been specified within AutoMed's Model Definitions Repository, including modelling languages for semi-structured data [3,18].

In our simple relational model, there are two kinds of schema construct: **Rel** and **Att**. The extent of a **Rel** construct  $\langle\langle R \rangle\rangle$  is the projection of relation  $R$  onto its primary key attributes  $k_1, \dots, k_n$ . The extent of each **Att** construct  $\langle\langle R, a \rangle\rangle$  where  $a$  is a non-key attribute of  $R$  is the projection of  $R$  onto  $k_1, \dots, k_n, a$ . We refer the reader to [13] for an encoding of a richer relational data model, including the modelling of constraints.

Suppose that  $\text{MAtab}(\underline{\text{CID}}, \underline{\text{SID}}, \text{Mark})$  and  $\text{IStab}(\underline{\text{CID}}, \underline{\text{SID}}, \text{Mark})$  are two source relations for a data warehouse respectively storing students' marks for two departments **MA** and **IS**, in which **CID** and **SID** are the course and student IDs. Suppose also that a relation  $\text{Course}(\underline{\text{Dept}}, \underline{\text{CID}}, \text{Avg})$  is in the data warehouse which gives the average mark for each course of each department.

The following transformation pathway expresses the schema transformation and integration processes in this example. Due to space limitations, we have not given the steps for removing the source relation constructs (note that this 'growing' and 'shrinking' of schemas is characteristic of AutoMed schema transformation pathways). Schema constructs  $\langle\langle \text{Details} \rangle\rangle$  and  $\langle\langle \text{Details}, \text{Mark} \rangle\rangle$  are temporary ones which are created for integrating the source data and then deleted after the global relation is created.

```
addRel <<Details>>      [{ 'MA' , k1 , k2 } | {k1 , k2} ← <<MAtab>>]
                        ++[ { 'IS' , k1 , k2 } | {k1 , k2} ← <<IStab>>];
addAtt <<Details, Mark>> [ { 'MA' , k1 , k2 , x } | {k1 , k2 , x} ← <<MAtab, Mark>>]
                        ++[ { 'IS' , k1 , k2 , x } | {k1 , k2 , x} ← <<IStab, Mark>>];
addRel <<Course>>      distinct [ {k , k1 } | {k , k1 , k2} ← <<Details>>];
addAtt <<Course, Avg>> [ {x , y , z } | {x , y} , z} ← (gc avg
                        [ { {k , k1} , x } | {k , k1 , k2 , x} ← <<Details, Mark>> ]]);
delAtt <<Details, Mark>> [ { 'MA' , k1 , k2 , x } | {k1 , k2 , x} ← <<MAtab, Mark>>]
                        ++[ { 'IS' , k1 , k2 , x } | {k1 , k2 , x} ← <<IStab, Mark>>];
delRel <<Details>>      [ { 'MA' , k1 , k2 } | {k1 , k2} ← <<MAtab>>]
                        ++[ { 'IS' , k1 , k2 } | {k1 , k2} ← <<IStab>>];
...

```

Note that some of the queries appearing in the above transformation steps are not SIQL but general IQL queries. In such cases, for the purposes of IVM, we decom-

pose a general IQL query into a sequence of SIQL queries by means of a depth-first traversal of the IQL query tree. For example, the IQL query  $[\{x, y, z\} | \{x, y\}, z] \leftarrow (\text{gc avg} [\{k, k1\}, x] | \{k, k1, k2, x\} \leftarrow \langle\langle \text{Details}, \text{Mark} \rangle\rangle])$  is decomposed into following sequence of SIQL queries, where  $v1$  and  $v2$  are virtual intermediate views:

$$v1 = \text{map} (\lambda \{k, k1, k2, x\}. \{k1, k2\}, x) \langle\langle \text{Details}, \text{Mark} \rangle\rangle$$

$$v2 = \text{gc avg } v1$$

$$v = \text{map} (\lambda \{x, y\}, z. \{x, y, z\}) v2$$

From now on, we assume that all queries in transformation steps are SIQL queries.

### 3 IVM with AutoMed Schema Transformations

Our IVM algorithms use the individual steps of a transformation pathway to compute the changes to each intermediate construct in the pathway, and finally obtain the changes to the view created by the transformation pathway in a stepwise fashion. Since no construct in a global schema is contributed by **delete** and **contract** transformations, we ignore these transformations in our IVM algorithms. In addition, computing changes based on a transformation  $\text{rename}(O', O)$  is simple — the changes to  $O$  are the same as the changes to  $O'$ . Thus, we only consider **add** transformations here.

We can express a single **add** transformation step as an expression  $v = \mathcal{Q}(D)$ , in which  $v$  is the schema construct created by the transformation and  $\mathcal{Q}$  is the SIQL query over the data source  $D$ . In order to incrementally maintain the global schema data, we develop a set of IVM formulae for each SIQL query, and apply these IVM formulae on each transformation step to compute the changes to the construct created by the step. By following all the steps in the transformation pathway, we compute the intermediate changes step by step, finally ending up with the final changes to the global schema data.

#### 3.1 IVM Formulae for SIQL Queries

We use  $\Delta\mathcal{C}/\nabla\mathcal{C}$  to denote a collection of data items inserted into/deleted from a collection  $\mathcal{C}$ . There might be many possible expressions for  $\Delta\mathcal{C}$  and  $\nabla\mathcal{C}$  but not all are equally desirable. For example, we could simply let  $\nabla\mathcal{C} = \mathcal{C}$  and  $\Delta\mathcal{C} = \Delta\mathcal{C}^{new}$ , but this is equivalent to recomputing the view from scratch [16]. In order to avoid such definitions, we use the concept of *minimality* [9] to ensure that no unnecessary data are produced.

**Minimality Conditions.** Any changes ( $\Delta\mathcal{C}/\nabla\mathcal{C}$ ) to a data collection  $\mathcal{C}$ , including the data source and the view, must satisfy the following minimality conditions:

- (i)  $\nabla\mathcal{C} \subseteq \mathcal{C}$ : We only delete tuples that are in  $\mathcal{C}$ ;
- (ii)  $\Delta\mathcal{C} \cap \nabla\mathcal{C} = \emptyset$ : We do not delete a tuple and then reinsert it.

We now give the IVM formulae for each SIQL query, in which  $v$  denotes the view,  $D$  denotes the data sources,  $\Delta v/\nabla v$  and  $\Delta D/\nabla D$  denote the collections inserted into/deleted from  $v$  and  $D$ , and  $D^{new}$  denotes the source collect  $D$  after the update. We observe that these formulae guarantee that the above minimality conditions are satisfied of  $\Delta v$  and  $\nabla v$  provided they are satisfied by  $\Delta D$  and  $\nabla D$ .

1. *IVM formulae for distinct, map, and aggregate functions;*

Table 1 illustrates the IVM formulae for these functions. We can see that the IVM formulae for **distinct/max/min/avg** function require accessing the post-update data



**Table 1.** IVM formulae for distinct, map, and aggregate functions

$v$	$\Delta v$	$\nabla v$
distinct $D$	distinct $[x x \leftarrow \Delta D;$ not(member $v x)$ ]	distinct $[x x \leftarrow \nabla D;$ not(member $D^{new} x)$ ]
map $\lambda e1.e2 D$	map $\lambda e1.e2 \Delta D$	map $\lambda e1.e2 \nabla D$
max $D$	let $r1 = \max \Delta D; r2 = \max \nabla D$	
	$\begin{cases} \max \Delta D, & \text{if } (v < r1); \\ \emptyset, & \text{if } (v \geq r1) \& (v \neq r2); \\ \max D^{new}, & \text{if } (v > r1) \& (v = r2). \end{cases}$	$\begin{cases} v, & \text{if } (v < r1); \\ \emptyset, & \text{if } (v \geq r1) \& (v \neq r2); \\ v, & \text{if } (v > r1) \& (v = r2). \end{cases}$
min $D$	let $r1 = \min \Delta D; r2 = \min \nabla D$	
	$\begin{cases} \min \Delta D, & \text{if } (v > r1); \\ \emptyset, & \text{if } (v \leq r1) \& (v \neq r2); \\ \min D^{new}, & \text{if } (v < r1) \& (v = r2). \end{cases}$	$\begin{cases} v, & \text{if } (v > r1); \\ \emptyset, & \text{if } (v \leq r1) \& (v \neq r2); \\ v, & \text{if } (v < r1) \& (v = r2). \end{cases}$
count $D$	$v + (\text{count } \Delta D) - (\text{count } \nabla D)$	$v$
sum $D$	$v + (\text{sum } \Delta D) - (\text{sum } \nabla D)$	$v$
avg $D$	avg $D^{new}$	$v$

**Table 2.** IVM formulae for bag union and monus

$v$	$\Delta v$	$\nabla v$
$D1 ++ D2$	$(\Delta D1 -- \nabla D2) ++ (\Delta D2 -- \nabla D1)$	$(\nabla D1 -- \Delta D2) ++ (\nabla D2 -- \Delta D1)$
$D1 -- D2$	$((\Delta D1 -- \Delta D2) ++ (\nabla D2 -- \nabla D1))$ $-- (D2 -- D1)$	$((\nabla D1 -- \nabla D2) ++ (\Delta D2 -- \Delta D1))$ $\cap v$

source and using the view data; the formulae for **count/sum** function need to use the view data; and the formulae for **map** function only use the updates to the data source.

### 2. IVM formulae for grouping functions such as $\text{group } D$ and $\text{gc } f D$ ;

Grouping functions group a bag of pairs  $D$  on their first component, and may apply an aggregate function  $f$  to the second component. For the IVM of a view defined by a grouping function, we firstly find the data items in  $D$ , which are in the same groups of the updates, *i.e.* have the same first component with the updates. Then this smaller data collection can be used to compute the changes to the view, so as to save time and space overheads. For example, the IVM formulae for  $v = \text{gc } f D$  are as follows:

$$\Delta v = \text{gc } f \left[ \{x, y\} | \{x, y\} \leftarrow D^{new}; \text{member } [p|\{p, q\} \leftarrow (\Delta D ++ \nabla D)] x \right]$$

$$\nabla v = \{x, y\} | \{x, y\} \leftarrow v; \text{member } [p|\{p, q\} \leftarrow (\Delta D ++ \nabla D)] x$$

The IVM formulae for grouping functions require accessing the updated data source and using the view data.

### 3. IVM formulae for bag union and monus;

Table 2 illustrates IVM formulae for bag union and monus (see [9]), in which  $\cap$  is an intersection operator with the following semantics:  $D1 \cap D2 = D1 -- (D1 -- D2) = D2 -- (D2 -- D1)$ . The IVM formulae for bag union only use the changes to the data sources, while the formulae for bag monus have to use the view data and require an auxiliary view  $D2 -- D1$ . This auxiliary view is similarly incrementally maintained by using the IVM formulae for bag monus with  $D1 -- D2$ .

```

Algorithm IVM4Comp()
Begin:
    tempView = D1new;
    Δv = ΔD1;
    ∇v = ∇D1;
    for i = 2 to n, do
        if (ΔDi or ∇Di is not empty)
            tempView = tempView ⋈ ... ⋈c(i-1) D(i-1)new;
            ∇v = (∇v ⋈ci Dinew -- ∇v ⋈ci ΔDi) ++ ∇v ⋈ci ∇Di
                ++ (tempView ⋈ci ∇Di -- Δv ⋈ci ∇Di);
            Δv = (Δv ⋈ci Dinew -- Δv ⋈ci ΔDi) ++ tempView ⋈ci ΔDi;
        else
            Δv = Δv ⋈ci Dinew;
            ∇v = ∇v ⋈ci Dinew;
    return Δv and ∇v;
End

```

Fig. 1. The IVM4Comp Algorithm

#### 4. IVM formulae for comprehension $[\overline{x}|\overline{x}_1 \leftarrow D1; \dots; \overline{x}_n \leftarrow Dn; C_1; C_2; \dots; C_k]$ ;

For ease of discussion, we use the join operator  $\bowtie$  to express this comprehension. In particular,  $(D1 \bowtie_c D2) = [\{x, y\} | x \leftarrow D1; y \leftarrow D2; c]$  where  $c = C_1; \dots; C_k$ . More generally,  $(D1 \bowtie_{c_1, c_2} D2 \bowtie_{c_3} \dots \bowtie_{c_n} Dn) = [\overline{x}|\overline{x}_1 \leftarrow D1; \dots; \overline{x}_n \leftarrow Dn; c_1; c_2; \dots; c_n]$  in which  $c_i$  is the conjunction of those predicates from  $C_1, \dots, C_k$  which contain variables appearing in  $\overline{x}_i$  but without any variable appearing in  $\overline{x}_j, j > i$ .

We firstly give the IVM formulae of a view  $v = D1 \bowtie_c D2$  as follows. These IVM formulae can be derived from the propagation rules described in [9,5].

$$\begin{aligned} \Delta v &= (\Delta D_1 \bowtie_c D_2^{new} -- \Delta D_1 \bowtie_c \Delta D_2) ++ D_1^{new} \bowtie_c \Delta D_2 \\ \nabla v &= (\nabla D_1 \bowtie_c D_2^{new} -- \nabla D_1 \bowtie_c \Delta D_2) ++ \nabla D_1 \bowtie_c \nabla D_2 \\ &\quad ++ (D_1^{new} \bowtie_c \nabla D_2 -- \Delta D_1 \bowtie_c \nabla D_2) \end{aligned}$$

Then, the IVM algorithm, IVM4Comp, for incrementally maintaining the view  $v = (D1 \bowtie_{c_1, c_2} D2 \bowtie_{c_3} \dots \bowtie_{c_n} Dn)$  is given in Figure 1. This IVM algorithm for the comprehension needs to access all the post-update data sources.

The IVM4Comp algorithm is similar to the IVM algorithms discussed in [21] and [1], *i.e.* the Strobe and SWEEP algorithms, in the context of maintaining a multi-source data warehouse. Both the Strobe and the SWEEP algorithm perform an IVM procedure for each update to a data source so as to ensure the data warehouse is consistent with the updated data source. For both algorithms, the cost of the messaging between the data warehouse and the data sources for each update is  $O(n)$  where  $n$  is the number of data sources. However, in practice, warehouse data are normally long-term and just refreshed periodically. Our IVM4Comp algorithm is able to handle a batch of updates and is specifically designed for a periodic view maintenance policy. The message cost of our algorithm for a batch of updates to any of the data sources is  $O(n)$ .

#### 5. IVM formulae for member and not member functions;

For ease of discussion, we use  $\wedge$  and  $\bar{\wedge}$  to denote expressions with member and not member functions, *i.e.*  $D1 \wedge D2 = [x | x \leftarrow D1; \text{member } D2 \ x]$  and  $D1 \bar{\wedge} D2 =$

$[x|x \leftarrow D1; \text{not}(\text{member } D2 \ x)]$ . The IVM formulae for these two functions are given below, in which the function `countNum a D` returns the number of occurrences of the data item  $a$  in  $D$ , *i.e.*  $\text{countNum } a \ D = \text{count } [x|x \leftarrow D; x=a]$ . We can see that all post-update data sources are required in the IVM formulae.

```

v = [x|x ← D1; member D2 x]
  let r1 = [x|x ← ΔD2; (countNum x ΔD2) = (countNum x D2new)]
      r2 = ∇D2 ∧ D2new
      Δv = (ΔD1 ∧ D2new -- ΔD1 ∧ r1) ++ D1new ∧ r1
      ∇v = (∇D1 ∧ D2new -- ∇D1 ∧ r1) ++ (D1new ∧ r2 -- ΔD1 ∧ r2) ++ ∇D1 ∧ r2
v = [x|x ← D1; not(member D2 x)]
  let r1 = [x|x ← ΔD2; (countNum x ΔD2) = (countNum x D2new)]
      r2 = ∇D2 ∧ D2new
      Δv = (ΔD1 ∧ D2new -- ΔD1 ∧ r2) ++ D1new ∧ r2
      ∇v = (∇D1 ∧ D2new -- ∇D1 ∧ r2) ++ (D1new ∧ r1 -- ΔD1 ∧ r1) ++ ∇D1 ∧ r1

```

### 3.2 IVM for Schema Transformation Pathways

Having defined the IVM formulae for each SIQL query, the update to a construct created by a single `add` transformation step is obtained by applying the appropriate formulae to the step's query. Our IVM procedure for a single transformation step is `IVM4AStep(cd, ts)` and its output is the change to the construct created by step  $ts$  based on the changes  $cd$  to  $ts$ 's data sources. After obtaining the change to all the constructs created by a transformation pathway, the view created by the transformation pathway is incrementally maintained.

However, as discussed above, the post-update data sources and the view itself are required by some IVM formulae. In a general transformation pathway, some intermediate constructs might be virtual. If a required data collection is unavailable, *i.e.* not materialised, the `IVM4AStep` procedure cannot be applied.

Thus, in order to apply the `IVM4AStep` procedure along a transformation pathway, we have to precheck each `add` transformation in the pathway. If a virtual data collection is required by the IVM formula for a transformation step, we must firstly recover this data collection and store it in the data warehouse. This precheck only needs to be performed once for each transformation pathway in a data warehouse, unless the transformation pathway evolves due to the evolution of a data source schema. This materialisation increases the storage overhead of the data warehouse, but does not increase the message cost of the IVM process since these materialised constructs are also maintainable by using the IVM process along the transformation pathway.

Alternatively, we can use AutoMed's Global Query Processor (GQP) to evaluate the extent of a virtual construct during the IVM process so as to avoid increasing persistent storage overheads. However, since it is based on post-update data sources, AutoMed's GQP can only recover a post-update view. If a view is used in an IVM formula, this means the view is *before* the update, which cannot be recovered by AutoMed's GQP.

We now give an example of prechecking a transformation pathway. In Section 2.2, the transformation pathway generating the construct  $\langle\langle \text{Course}, \text{Avg} \rangle\rangle$  in the global schema can be expressed as the following sequence of view definitions, where the intermediate constructs  $\forall 1, \dots, \forall 4$  and  $\langle\langle \text{Details}, \text{Mark} \rangle\rangle$  are virtual:

```

v1           = [{ 'IS' , k1 , k2 , x } | {k1 , k2 , x} ← ⟨⟨IStab, Mark⟩⟩]
v2           = [{ 'MA' , k1 , k2 , x } | {k1 , k2 , x} ← ⟨⟨MAtab, Mark⟩⟩]
⟨⟨Details, Mark⟩⟩ = v1 ++ v2
v3           = map (λ{k , k1 , k2 , x} . { {k , k1} , x }) ⟨⟨Details, Mark⟩⟩
v4           = gc avg v3
⟨⟨Course, Avg⟩⟩  = map (λ{x , y , z} . {x , y , z}) v4

```

In order to incrementally maintain ⟨⟨Course, Avg⟩⟩, the intermediate views  $v_3$  and  $v_4$  must be materialised (based on the IVM formulae for grouping functions). For example, supposes that an update to the data sources is a tuple inserted into ⟨⟨IStab, Mark⟩⟩,  $\Delta\langle\langle\text{IStab, Mark}\rangle\rangle = \{ 'ISC01' , 'ISS05' , 80 \}$ . Following on the transformation pathway, we obtain the changes to the intermediate views as follows:

```

Δv1           = { 'IS' , 'ISC01' , 'ISS05' , 80 }
Δ⟨⟨Details, Mark⟩⟩ = { 'IS' , 'ISC01' , 'ISS05' , 80 }
Δv3           = { 'IS' , 'ISC01' , 80 }

```

Since the extents of  $v_3$  and  $v_4$  are recovered, changes to  $v_4$  can be obtained by using the IVM formulae for grouping functions, and then be used to compute changes to ⟨⟨Course, Avg⟩⟩ by using the IVM formulae for `map`.

However, the post-update extent of  $v_3$  can be recovered by AutoMed's GQP, and using the inverse query of `map (λ{x , y , z} . {x , y , z}) v4`, the pre-update extent of  $v_4$  can also be recovered as  $v_4 = \text{map} (\lambda\{x , y , z\} . \{x , y , z\}) \langle\langle\text{Course, Avg}\rangle\rangle$ . Thus, in practice, no intermediate view needs to be materialised for incrementally maintaining ⟨⟨Course, Avg⟩⟩ along the pathway. In the future, we will investigate these avoidable materializations more generally, so as to apply them in our IVM algorithms.

## 4 Concluding Remarks

AutoMed schema transformation pathways can be used to express data transformation and integration processes in heterogeneous data warehousing environments. This paper has discussed techniques for incremental view maintenance along such pathways and thus addresses the general IVM problem for heterogeneous data warehouses. We have developed a set of IVM formulae. Based on these formulae, our algorithms perform an IVM process along a schema transformation pathway. We are currently implementing the algorithms as part of a broader bioinformatics data warehousing project (BIOMAP).

One of the advantages of AutoMed is that its schema transformation pathways can be readily evolved as the data warehouse evolves [8]. In this paper we have shown how to perform IVM along such evolvable pathways.

Although this paper has used IQL as the query language in which transformations are specified, our algorithms are not limited to one specific data model or query language, and could be applied to other query languages involving common algebraic operations such as selection, projection, join, aggregation, union and difference.

Finally, since our algorithms consider in turn each transformation step in a transformation pathway in order to compute data changes in a stepwise fashion, they are useful not only in data warehousing environments, but also in any data transformation and integration framework based on sequences of primitive schema transformations. For example, Zamboulis and Poulouvasilis present an approach for integrating heteroge-

neous XML documents using the AutoMed toolkit [18,19]. A schema is automatically extracted for each XML document and transformation pathways are applied to these schemas. McBrien and Poulouvasilis also discusses how AutoMed can be applied in peer-to-peer data integration settings [14]. Thus, the IVM approach we have discussed in this paper is readily applicable in peer-to-peer and semi-structured data integration environments.

## References

1. D. Agrawal, A. E. Abbadi, A. K. Singh, and T. Yurek. Efficient view maintenance at data warehouses. In *Proc. ACM SIGMOD'97*, pages 417–427. ACM Press, 1997.
2. J. Albert. Algebraic properties of bag data types. In *Proc. VLDB'91*, pages 211–219, 1991.
3. M. Boyd, S. Kittivoravitkul, and C. Lazanitis. AutoMed: A BAV data integration system for heterogeneous data sources. In *Proc. CAiSE'04*, LNCS 3084, 2004.
4. P. Buneman *et al.* Comprehension syntax. *SIGMOD Record*, 23(1):87–96, 1994.
5. L. S. Colby, T. Griffin, L. Libkin, I. S. Mumick, and H. Trickey. Algorithms for deferred view maintenance. In *Proc. ACM SIGMOD'96*, pages 469–480, 1996.
6. G. Dong. Incremental maintenance of recursive views: A survey. In *Materialized Views: Techniques, Implementations, and Applications*, pages 159–162. The MIT Press, 1999.
7. H. Fan and A. Poulouvasilis. Using AutoMed metadata in data warehousing environments. In *Proc. DOLAP'03*, pages 86–93. ACM Press, 2003.
8. H. Fan and A. Poulouvasilis. Schema evolution in data warehousing environments — a schema transformation-based approach. In *Proc. ER'04*, LNCS, pages 639–653, 2004.
9. T. Griffin and L. Libkin. Incremental maintenance of views with duplicates. In *Proc. ACM SIGMOD'95*, pages 328–339. ACM Press, 1995.
10. A. Gupta and I. S. Mumick. Maintenance polices. In *Materialized Views: Techniques, Implementations, and Applications*, pages 9–11. The MIT Press, 1999.
11. Ashish Gupta, H. V. Jagadish, and Inderpal Singh Mumick. Data integration using self-maintainable views. In *Extending Database Technology*, pages 140–144, 1996.
12. E. Jasper, A. Poulouvasilis, and L. Zamboulis. Processing IQL queries and migrating data in the AutoMed toolkit. Technical Report 20, Automed Project, 2003.
13. P. McBrien and A. Poulouvasilis. A uniform approach to inter-model transformations. In *Proc. CAiSE'99*, volume 1626 of LNCS, pages 333–348. Springer, 1999.
14. P. McBrien and A. Poulouvasilis. Defining peer-to-peer data integration using both as view rules. In *Proc. DBISP2P, Berlin, Germany, September 7-8*, LNCS. Springer, 2003.
15. A. Poulouvasilis. A Tutorial on the IQL Query Language. Technical Report 28, Automed Project, 2004.
16. D. Quass. Maintenance expressions for views with aggregation. In *Proc. VIEW'96*, pages 110–118, 1996.
17. D. Quass, A. Gupta, I.S. Mumick, and J. Widom. Making views self-maintainable for data warehousing. In *Proc. PDIS'96*, pages 158–169, 1996.
18. L. Zamboulis. XML data integration by graph restructuring. In *Proc. BNCOD'04*, volume 3112 of LNCS, pages 57–71. Springer-Verlag, 2004.
19. L. Zamboulis and A. Poulouvasilis. Using AutoMed for XML data transformation and integration. In *Proc. DIWeb'04*, pages 58–69, 2004.
20. Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom. View maintenance in a warehousing environment. In *Proc. ACM SIGMOD'95*, pages 316–327, 1995.
21. Yue Zhuge, Hector Garcia-Molina, and Janet L. Wiener. Consistency algorithms for multi-source warehouse view maintenance. *Distributed and Parallel Databases*, 6(1):7–40, 1998.

# Data Mapper: An Operator for Expressing One-to-Many Data Transformations

Paulo Carreira<sup>1</sup>, Helena Galhardas<sup>2</sup>, João Pereira<sup>2</sup>, and Antónia Lopes<sup>1</sup>

<sup>1</sup> Faculty of Sciences of the University of Lisbon,  
C6 - Piso 3, 1700 Lisboa, Portugal

paulo.carreira@oblog.pt\*, mal@di.fc.ul.pt

<sup>2</sup> INESC-ID and Instituto Superior Técnico, Avenida Prof. Cavaco Silva,  
Tagus Park, 2780-990 Porto Salvo, Portugal  
hig@inesc-id.pt, joao@inesc-id.pt

**Abstract.** Transforming data is a fundamental operation in application scenarios involving *data integration*, *legacy data migration*, *data cleaning*, and *extract-transform-load processes*. Data transformations are often implemented as relational queries that aim at leveraging the optimization capabilities of most RDBMSs. However, relational query languages like SQL are not expressive enough to specify an important class of data transformations that produce several output tuples for a single input tuple. This class of data transformations is required for solving the data heterogeneities that occur when source data represents an aggregation of target data.

In this paper, we propose and formally define the *data mapper operator* as an extension of the relational algebra to address one-to-many data transformations. We supply an algebraic rewriting technique that enables the optimization of data transformation expressions that combine filters expressed as standard relational operators with mappers. Furthermore, we identify the two main factors that influence the expected optimization gains.

## 1 Introduction

In general, data transformations aim at integrating data from multiple sources, migrating legacy data, cleaning data or performing ETL processes that support data warehousing. When putting in place such initiatives, data represented by a fixed source schema must be transformed into a fixed target data schema. In this context, we frequently face the problem of *data heterogeneities*, i.e., the use of different representations of the same data in source and target schemas [14,11]. Several factors cause the existence of data heterogeneities, for example: (i) different units of measurement, (ii) different representations of compound data (e.g. multiple attributes representing day, month and year information *vs* a single date attribute), or (iii) distinct representations of the same data domain (e.g. {true, false} *vs* {yes, no} for boolean values). Another important source of

---

\* The author's work was partially supported by OBLOG Consulting S.A.

data heterogeneities is the representation of data according to different aggregation levels (e.g. hourly *vs* daily). When the source data represents an aggregation of the target data (e.g., yearly aggregated data in the source and monthly data in the target), the data transformation that has to take place needs to generate several output tuples for each input tuple. This class of data transformations will be henceforth designated as *one-to-many* data transformations.

Relational algebra (RA) expressions are often used to implement data transformations. In fact, simple data transformations can be naturally expressed as RA queries. Moreover, data often resides in RDBMSs and data transformations specified as relational expressions can take direct advantage of their optimization capabilities. However, due to the limitations in the expressive power of RA [1], relational queries are insufficient for expressing many interesting data transformations [12,13].

The normalization theory underlying the relational model imposes the organization of data according to several relations in order to avoid duplication and inconsistency of information. Therefore, data retrieved from the database is mostly obtained by selecting, joining and unioning relations. Data transformation applications bring a new requirement to RA as their focus is no more limited to the idea of selecting information [1] but also involves the production of new data items. In particular, RA is not powerful enough to represent one-to-many data transformations.

In this paper, we propose to extend the RA with the *mapper* operator, which significantly increases its expressive power by representing one-to-many data transformations. Informally, a mapper is applied to an input relation and produces an output relation. It iterates over each input tuple and generates one or more output tuples, by applying a set of domain-specific functions. This way, it supports the dynamic creation of tuples based on a source tuple contents. This kind of operation also appears implicitly in most languages aiming at implementing schema and data transformations but, as far as we know, has never been properly handled as a first-class operator. Considering the mapper operator an extension to the RA brings new optimization opportunities as shown in [5].

The main contributions of this paper are the following:

- a formal definition of a new primitive operator, named *data mapper*, that allows expressing one-to-many data mappings;
- an algebraic rewriting technique for optimizing expressions involving mappers and filters expressed as relational operators;
- the identification of two important factors that influence the gains obtained when applying the proposed optimization.

The remainder of the paper is organized as follows. The formalization of the mapper is presented in Section 2. Section 3 presents the algebraic rewriting technique that enables the optimization of expressions involving the mapper operator. Practical evidence that shows the advantage of the optimization is presented in Section 4. Finally, related work is summarized in Section 5 and conclusions are presented in Section 6.

## 1.1 Motivation

As mentioned above, there is a considerable amount of data transformations that require one-to-many data mappings. In this section, we motivate the reader by presenting an example based on a real-world legacy-data migration scenario, that has been intentionally simplified for illustration purposes.

Relation LOANS		Relation PAYMENTS		
ACCT	AM	ACCTNO	AMOUNT	SEQNO
12	20.00	0012	20.00	1
3456	140.00	3456	100.00	1
901	250.00	3456	40.00	2
		0901	100.00	1
		0901	100.00	2
		0901	50.00	3

**Fig. 1.** (a) On the left, the LOANS relation and, (b) on the right, the PAYMENTS relation

*Example 1.* Consider the source relation  $\text{LOANS}[\text{ACCT}, \text{AM}]$  (represented in Figure 1) that stores the details of loans requested per account. Suppose LOANS data must be transformed into  $\text{PAYMENTS}[\text{ACCTNO}, \text{AMOUNT}, \text{SEQNO}]$ , the target relation, according to the following requirements:

1. In the target relation, all the account numbers are left padded with zeroes. Thus, the attribute **ACCTNO** is obtained by (left) concatenating zeroes to the value of **ACCT**.
2. The target system does not support payment amounts greater than 100. The attribute **AMOUNT** is obtained by breaking down the value of **AM** into multiple parcels with a maximum value of 100, in such a way that the sum of amounts for the same **ACCTNO** is equal to the source amount for the same account. Furthermore, the target field **SEQNO** is a sequence number for the parcel. This sequence number starts at 1 for each sequence of parcels of a given account.

The implementation of data transformations similar to those requested for producing the target relation **PAYMENTS** of Example 1 is challenging, since solutions to the problem involve the dynamic creation of tuples based on the value of attribute **AM**.

## 2 The Mapper Operator

We start by introducing some preliminary notation. Let  $R(A_1, \dots, A_n)$  be a *relation schema*. The domain of the relation schema  $R$  is represented as  $\text{Dom}(A)$ , where  $A$  represents the relation schema composed by the attributes  $A_1, \dots, A_n$ . A *relation instance* (or relation, for short) of  $R(A)$  is written as  $r(A)$  or simply  $r$ . Each element  $t$  of  $r$  is called a *tuple* or *r-tuple* and can be regarded as a function that associates a value of  $\text{Dom}(A_i)$  to each  $A_i \in A$ ; we denote this value by



$t[A_i]$ . Given a set of distinct attributes  $B = \{B_1, \dots, B_k\}$  where each  $B_i \in A$ , we denote by  $t[B]$ , the tuple  $\langle t[B_1], \dots, t[B_k] \rangle$ . We assume two fixed relation schemas  $S(X_1, \dots, X_n)$  and  $T(Y_1, \dots, Y_m)$ . We refer to  $S$  and  $T$  as the source and the target relation schemas, respectively.

A mapper is a unary operator  $\mu_F$  that takes a relation instance of the source relation schema as input and produces a relation instance of the target relation schema as output. The mapper operator is parameterized by a set  $F$  of special functions, which we designate as *mapper functions*.

Roughly speaking, each mapper function allows one to express a part of the envisaged data transformation, focused on one or more attributes of the target schema. Although the idea is to apply mapper functions to tuples of a source relation instance, it may happen that some of the attributes of the source schema are irrelevant for the envisaged data transformation. The explicit identification of the attributes that are considered relevant is then an important part of a mapper function. Mapper functions are formally defined as follows.

**Definition 1.** *Let  $A$  be a non-empty list of distinct attributes in  $Y_1, \dots, Y_m$ . An  $A$ -mapper function for transforming the data of  $S$  into  $T$  consists of a non-empty list of distinct attributes  $B$  in  $X_1, \dots, X_n$  and a computable function  $f_A: Dom(B) \rightarrow \mathcal{P}(Dom(A))$ .*

*Let  $t$  be tuple of a relation instance of the source schema. We define  $f_A(t)$  to be the application of the underlying function  $f_A$  to the tuple  $t$ , i.e.,  $f_A(t[B])$ .*

We shall freely use  $f_A$  to denote both the mapper function and the function itself. In this way, mapper functions describe how a specific part of the target data can be obtained from the source data. The intuition is that each mapper function establishes how the values of a group of attributes of the target schema can be obtained from the attributes of the source schema. Another key point is that, when applied to a tuple, a mapper function can produce a set of values of  $Dom(f_A)$ , rather than a single value.

As mentioned before, a mapper operator is parameterized by a set of mapper functions. This set is said to be *proper* for transforming the data from the source to the target schemas if it specifies, in a unique way, how the values of every attribute of the target schema are produced.

**Definition 2.** *A set  $F = \{f_{A_1}, \dots, f_{A_k}\}$  of mapper functions is said to be proper (for transforming the data of  $S$  into  $T$ ) iff every attribute  $Y_i$  of the target relation schema is an element of exactly one of the  $A_j$  lists, for  $1 \leq j \leq k$ .*

The mapper operator  $\mu_F$  puts together the data transformations of the input relation defined by the mapper functions in  $F$ . Given a tuple  $s$  of the input relation,  $\mu_F(s)$  consists of the tuples  $t$  in  $Dom(Y)$  such that, for every  $i$ , to the attributes in  $A_i$ , associate the values given by  $f_{A_i}(s)$ . Formally, the mapper operator is defined as follows.

**Definition 3.** *Given a relation  $s(X)$  and a proper set of mapper functions  $F = \{f_{A_1}, \dots, f_{A_k}\}$ , the mapper of  $s$  with respect to  $F$ , denoted by  $\mu_F(s)$ , is the relation instance of the target relation schema defined by*

$$\mu_F(s) \stackrel{\text{def}}{=} \{t \in \text{Dom}(Y) \mid \exists u \in s \text{ s.t. } t[A_i] \in f_{A_i}(u), \forall 1 \leq i \leq k\}$$

The semantics of the mapper operator can be given also as a Cartesian product of the results of each mapper function. Definition 3 is more abstract than the definition in terms of Cartesian product operations in the sense that it does not impose any particular ordering on the attributes. For further details about the properties and expressiveness of the mapper operator, please refer to [5]. In order to illustrate this new operator, we revisit Example 1.

*Example 2.* The requirements presented in Example 1 can be described by the mapper  $\mu_{acct,amt}$ , where *acct* is an ACCT-mapper function that returns a singleton with the account number ACCT properly left padded with zeroes and *amt* is the [AMOUNT,SEQNO]-mapper function s.t.,  $amt(am)$  is given by

$$\{(100, i) \mid 1 \leq i \leq (am/100)\} \cup \{(am\%100, (am/100) + 1) \mid am\%100 \neq 0\}$$

where we have used / and % to represent the integer division and modulus operations, respectively.

For instance, if  $t$  is the source tuple (901, 250.00), the result of evaluating  $amt(t)$  is the set  $\{(100, 1), (100, 2), (50, 3)\}$ . Given a source relation  $s$  including  $t$ , the result of the expression  $\mu_{acct,amt}(s)$  is a relation that contains the set of tuples  $\{\langle '0901', 100, 1 \rangle, \langle '0901', 100, 2 \rangle, \langle '0901', 50, 3 \rangle\}$ .

### 3 Optimization of Sequences of Filters and Mappers

In this section, we present a logical optimization technique that comprises two algebraic rewriting rules for optimizing expressions that combine filters expressed as relational selection operators with mappers. The first rule presented alleviates the cost of performing the Cartesian product operations that are used to implement the mapper operator. The second rule avoids superfluous function evaluations by pushing selections to the sources, thus reducing the number of tuples fed to the mapper as early as possible. Each rule is presented as an equivalence between terms. This equivalence is complemented by a text that describes the semantic pre-conditions that guarantee its correctness.

#### 3.1 Pushing Selections to Mapper Functions

When applying a selection to a mapper we can take advantage of the mapper semantics to introduce an important optimization. Given a selection  $\sigma_{C_{A_i}}$  applied to a mapper  $\mu_{f_{A_1}, \dots, f_{A_k}}$ , this optimization consists of pushing the selection  $\sigma_{C_{A_i}}$ , where  $C_{A_i}$  is a condition on the attributes produced by some mapper function  $f_{A_i}$ , directly to the output of the mapper function.

**RULE 1:** Let  $F = \{f_{A_1}, \dots, f_{A_k}\}$  be a set of multi-valued mapper functions, proper for transforming  $S(X)$  into  $T(Y)$ . Consider a condition  $C_{A_i}$  dependent on a list of attributes  $A_i$  such that  $f_{A_i} \in F$ . Then,

$$\sigma_{C_{A_i}}(\mu_F(s)) = \mu_{F \setminus \{f_{A_i}\} \cup \{\sigma_{C_{A_i}} \circ f_{A_i}\}}(s)$$

where  $(\sigma_{C_{A_i}} \circ f_{A_i})(t) = \{f_{A_i}(t) \mid C_{A_i}(t)\}$ .

Note that when  $C_{A_i}(t)$  does not hold, the function  $\sigma_{C_{A_i}}(f_{A_i})(t)$  returns the empty set. Whenever a mapper function  $f_{A_i}$  applied to a tuple  $t$  returns an empty set, the result of the mapper will also be an empty set. Hence, we may skip the evaluation of all mapper functions  $f_{A_j}$ , such that  $j \neq i$ . Physical execution algorithms for the mapper operator can take advantage of this optimization by evaluating  $f_{A_i}$  before any other mapper function<sup>1</sup>.

### 3.2 Pushing Selections Through Mappers

An alternative way of rewriting expressions of the form  $\sigma_C(\mu_F(s))$  consists of replacing the attributes that occur in the condition  $C$  with the mapper functions that compute them. Suppose that, in the selection condition  $C$ , attribute  $A$  is produced by the mapper function  $f_A$ . By replacing the attribute  $A$  with the mapper function  $f_A$  in condition  $C$  we obtain an equivalent condition.

In order to formalize this notion, we first need to introduce some notation. Let  $F = \{f_{A_1}, \dots, f_{A_k}\}$  be a set of mapper functions proper for transforming  $S(X)$  into  $T(Y)$ . The function resulting from the restriction of  $f_{A_i}$  to an attribute  $Y_j \in A_i$  is denoted by  $f_{A_i} \downarrow Y_j$ . Moreover, given an attribute  $Y_j \in Y$ ,  $F \downarrow Y_j$  represents the function  $f_{A_i} \downarrow Y_j$  s.t.  $Y_j \in A_i$ . Note that, because  $F$  is a proper set of mapper functions, the function  $F \downarrow Y_j$  exists and is unique.

**RULE 2:** Let  $F = \{f_{A_1}, \dots, f_{A_k}\}$  be a set of single-valued mapper functions, proper for transforming  $S(X)$  into  $T(Y)$ . Let  $B = B_1 \cdot \dots \cdot B_k$  be a list of attributes in  $Y$  and  $s$  a relation instance of  $S(X)$ . Then,

$$\sigma_{C_B}(\mu_F(s)) = \mu_F(\sigma_{C[B_1, \dots, B_k \leftarrow F \downarrow B_1, \dots, F \downarrow B_k]}(s))$$

where  $C_B$  means that  $C$  depends on the attributes of  $B$ , and the condition that results from replacing every occurrence of each  $B_i$  by  $E_i$  is represented as  $C[B_1, \dots, B_n \leftarrow E_1, \dots, E_n]$ .

This rule replaces each attribute  $B_i$  in the condition  $C$  by the expression that describes how its values are obtained.

## 4 Practical Validation

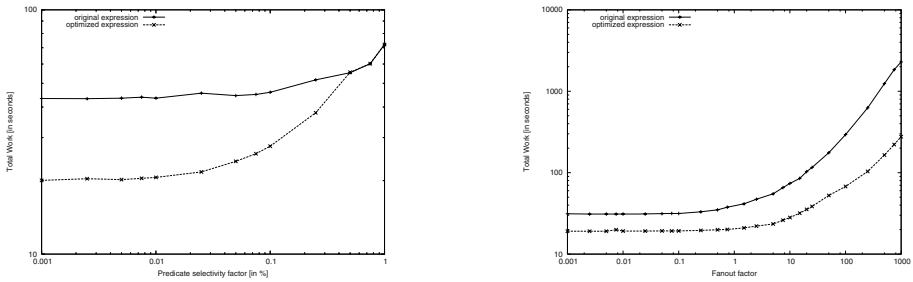
In order to validate the optimizations proposed in Section 3, we have implemented the mapper operator and conducted a number of experiments contrasting expressions consisting of mappers and selections with their optimized equivalents. Due to space limitations we only focus on the optimization obtained by applying Rule 1.

We have identified the *mapper function fanout* and the *predicate selectivity* [16], as two important factors that influence the gain obtained when applying

<sup>1</sup> This optimization can be generalized to first evaluate those functions with higher probability of yielding an empty set. Fundamentally, this is the same as finding the *optimal predicate ordering* addressed in [10].

the proposed optimization. Similarly to [6], we designate the average cardinality of the output values produced by a mapper function for each input tuple as function fanout. Our experiments only address the influence of the two above mentioned factors by employing cheap functions. Other factors that influence the mapper execution cost are the I/O cost and the function evaluation cost<sup>2</sup>.

The experiments apply a mapper  $\mu_{f_1, f_2, f_3, f_4}$  to synthetic data where, unless otherwise stated, each mapper function has a fanout factor of 2.0. In all experiments, we test the original expression  $\sigma_{p_i}(\mu_{f_1, f_2, f_3, f_4}(r))$  and the optimized expression  $\mu_{f_1, \sigma_{p_i} \circ f_2, f_3, f_4}(r)$ , where predicate  $p_i$  has some predefined selectivity and  $r$  is an input relation with a predefined size. Each experiment measures *total work*, i.e., the sum of the time taken to read the input tuples, to compute the output tuples, and to materialize them.



**Fig. 2.** Evolution of total work for the original and optimized expressions with 1 million tuples. (a) On the left, the effect of increasing selectivity factors for a mapper with four functions with a fanout factor of 2.0 (b) On the right, the effect of increasing mapper function fanout factors with a predicate selectivity fixed to 2.5%.

**The Influence of Predicate Selectivity.** To understand the effect of the predicate selectivity, a set of experiments was carried out using a different  $p_i$  predicate with selectivity factor ranging from 0.1% to 100%. The tests were executed over an input relation with 1 million input tuples. Figure 2a shows the evolution of the total work for different selectivities.

As expected, the highest gains brought by the optimization were obtained for smaller selectivity factors. More concretely, for a selectivity of 0.1%, the optimized expression was 2.16 times faster than the original one. As the selectivity factor decreases, more results are filtered out from function  $f_2$  and, therefore, the cost of computing the Cartesian product involved in the mapper is lower. As the selectivity tends to 100%, the gain drops since the results filtered out from  $f_2$  tend to 0%. Nevertheless, there is still, albeit small, a gain due to the reduction on the number of predicate evaluations (recall that each function has a fanout of 2). This gain is small since the cost of a predicate evaluation is, in our experiments, low.

<sup>2</sup> We do not take into account these factors in our experiments, since Rule 1 is not able to optimize them.

**The Influence of Function Fanout.** To understand the effects of the function fanout on the optimization proposed, we tracked the evolution of total work for the original and optimized expressions when the fanout factor varies. Function  $f_2$  was replaced by a function that guarantees a predefined fanout factor ranging from 0.001 (unusually small) to 1000. To isolate the effect of the fanout, the fanout factors of the remaining functions were set to 1.0 and the selectivity of the predicate was kept constant at 2.25%. The results are depicted in Figure 2b.

For small values of fanout, the gain brought by the optimization remains mostly constant (an improvement of  $\approx 61\%$ ). This is due to the fact that for low values of fanout, the cost of performing the Cartesian product is lower than the combined costs of I/O and evaluation of the mapper functions. Notice that the cost of the Cartesian product increases with the fanout, since the higher the fanout, the more tuples have to be produced by the Cartesian product for each input tuple. Thus, for high fanout values, the cost of performing the Cartesian product becomes the dominant factor. Hence, the gain of the optimization increases with the fanout factor since our optimization reduces the cost of the Cartesian product.

## 5 Related Work

Data transformation is an old problem and the idea of using a query language to specify such transformations has been proposed back in the 1970's with two prototypes, Convert [17] and Express [18], both aiming at data conversion. More recently, three efforts, Potter's Wheel [15], Ajax [7] and Data Fusion [3], have proposed operators for data transformation and cleaning purposes.

Potter's Wheel `fold` operator is capable of producing several output tuples for each input tuple. The main difference w.r.t. the mapper operator lies in the number of output tuples generated. In the case of the `fold` operator, the number of output tuples is bound to the number of columns of the input relation, while the mapper operator may generate an arbitrary number of output tuples.

The semantics of the Ajax `map` operator represents exactly a one-to-many mapping, but it has not been proposed as an extension of the relational algebra. Consequently, the issue of semantic optimization, as we propose in this paper, has not been addressed for the Ajax `map`. Data Fusion implements the semantics of the mapper operator as it is presented here. However, the current version of Data Fusion is not supported by an extended relational algebra as we propose.

Our decision of adopting database technology as a basis for data transformation is not completely revolutionary (see, e.g., [9,2]). Several RDBMSs, like Microsoft SQL Server, already include additional software packages specific for ETL tasks. However, to the best of our knowledge, none of these extensions is supported by the corresponding theoretical background in terms of existing database theory. Therefore, the capabilities of relational engines, for example, in terms of optimization opportunities are not fully exploited for ETL tasks.

Recently, [19] has proposed a rigorous approach to the problem of optimizing an ETL process defined as a workflow of data transformation activities. The authors model the ETL optimization problem as a global state-space search problem. In our approach, we use local optimization, since an ETL transformation program must be represented by a set of extended relational algebra expressions to be optimized one at a time.

A preliminary version of this work [4] has presented the idea of performing the logical optimization of queries involving mappers and standard relational operators. However, the formal aspects concerning the mapper operator and the optimization rules were not detailed. In [5], we analyzed the optimization of expressions that combine mappers with other relational algebra operators and presented the formal correctness proofs for the rules.

## 6 Conclusions and Future Work

In this paper, we have addressed the problem of specifying one-to-many data transformations that are frequently required in data integration, data cleaning, legacy-data migration, and ETL scenarios. Practical evidence gathered from the Ajax [8] data cleaning prototype and from Data Fusion [3] legacy-data migration tool, which has been used commercially in large legacy-data migration projects, corroborates the need of supporting data transformations that require one-to-many mappings. Since one-to-many data transformations are not expressible through standard RA queries, we proposed the mapper operator. This new operator allow us to naturally express one-to-many data transformations, while extending the expressive power of RA at the same time. We show in [5] that RA extended with the mapper operator is more expressive than standard RA.

We presented a simple formal semantics for the mapper operator that can be implemented using Cartesian product operations. We then introduced two provenly correct algebraic rewriting rules that aim at optimizing queries that combine standard relational filters and data transformations encoded as mappers. To assess the effectiveness of our approach, we have implemented the mapper operator and conducted a set of experiments for observing the behavior of the rewriting rule that consists of pushing selection conditions to the output of mapper functions. Our experiments indicate that substantial performance improvements are obtained by employing the proposed rewriting, even in the presence of cheap functions. Furthermore, we were able to isolate two important factors, the predicate selectivity and the mapper function fanout, which strongly influence the performance gains obtained.

Currently, we are developing and experimenting different physical execution algorithms for the mapper operator. This way, we intend to complement the logical optimization technique presented with physical optimizations that can be integrated into an RDBMS optimizer. We strongly believe that this solution will definitely contribute to the application of the current relational database technology for enhancing the performance of data transformation engines.

## References

1. A. V. Aho and J. D. Ullman. Universality of data retrieval languages. In *Proc. of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 110–119. ACM Press, 1979.
2. P. A. Bernstein and E. Rahm. Data warehouse scenarios for model management. In *Int'l Conf. on Conceptual Modeling / the Entity Relationship Approach*, 2000.
3. P. Carreira and H. Galhardas. Efficient development of data migration transformations. In *ACM SIGMOD Int'l Conf. on the Management of Data*, June 2004.
4. P. Carreira and H. Galhardas. Execution of Data Mappers. In *Int'l Workshop on Information Quality in Information Systems*. ACM, June 2004.
5. P. Carreira, H. Galhardas, A. Lopes, and J. Pereira. Extending the relational algebra with the Mapper operator. DI/FCUL TR 05-2, Department of Informatics, University of Lisbon, January 2005. Available at the url <http://www.di.fc.ul.pt/tech-reports>.
6. S. Chaudhuri and K. Shim. Query optimization in the presence of foreign functions. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'93)*, 1993.
7. H. Galhardas, D. Florescu, D. Shasha, and E. Simon. Ajax: An extensible data cleaning tool. *ACM SIGMOD Int'l Conf. on Management of Data*, 2(29), 2000.
8. H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001.
9. L. Haas, R. Miller, B. Niswonger, M. T. Roth, P.M. Schwarz, and E. L. Wimmers. Transforming heterogeneous data with database middleware: Beyond integration. *Special Issue on Data Transformations. IEEE Data Eng. Bulletin*, 22(1), 1999.
10. J. M. Hellerstein. Optimization techniques for queries with expensive methods. *ACM Transactions on Database Systems*, 22(2):113–157, June 1998.
11. W. Kim, B.-J. Choi, E.-K. Hong, S.-K. Kim, and D. Lee. A taxonomy of dirty data. *Data Mining and Knowledge Discovery*, 7(1):81–99, January 2003.
12. L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. SchemaSQL - A Language for Querying and Restructuring Database Systems. In *Proc. Int'l Conf. on Very Large Databases (VLDB'96)*, pages 239–250, Bombay, India, September 1996.
13. R. J. Miller. Using Schematically Heterogeneous Structures. *Proc. of ACM SIGMOD Int'l Conf. on the Management of Data*, 2(22):189–200, June 1998.
14. E. Rahm and H.-H. Do. Data Cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 24(4), 2000.
15. V. Raman and J. M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*, 2001.
16. P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *ACM SIGMOD Int'l Conf. on the Management of Data*, 1979.
17. N. C. Shu, B. C. Housel, and V. Y. Lum. CONVERT: A High Level Translation Definition Language for Data Conversion. *Communications of the ACM*, 18(10):557–567, October 1975.
18. N. C. Shu, B. C. Housel, R. W. Taylor, S. P. Ghosh, and V. Y. Lum. EXPRESS: A Data EXtraction, Processing and REStructuring System. *ACM Transactions on Database Systems*, 2(2):134–174, June 1977.
19. A. Simitis, P. Vassiliadis, and T. K. Sellis. Optimizing ETL processes in data warehouses. In *Proc. of the 21st Int'l Conf. on Data Engineering (ICDE)*, 2005.

# Parallel Consistency Maintenance of Materialized Views Using Referential Integrity Constraints in Data Warehouses

Jinho Kim<sup>1</sup>, Byung-Suk Lee<sup>1</sup>, Yang-Sae Moon<sup>1</sup>, Soo-Ho Ok<sup>2</sup>, and Wookey Lee<sup>3</sup>

<sup>1</sup> Dept. of Computer Science, Kangwon Nat'l University,  
192-1 Hyoja Dong 2, Chunchon, Kangwon, Korea  
{jhhkim, bsdream, ysmoon}@kangwon.ac.kr

<sup>2</sup> Dept. of Computer Science, Kosin University,  
149-1 Dongsam Dong, Youngdo Ku, Pusan, Korea  
shok@kosin.ac.kr

<sup>3</sup> Dept. of Computer Science, Sungkyul University,  
147-2 Anyang 8 Dong, Anyang, Kyungki, Korea  
wook@sungkyul.edu

**Abstract.** Data warehouses can be considered as materialized views which maintain the online analytical information extracted from distributed data sources. When data sources are changed, materialized views should be maintained correspondingly to keep the consistency between data sources and materialized views. If a view is defined through joining several source relations, an update in one source relation invokes a set of join subqueries thus the view maintenance takes much time of processing. In this paper, we propose a view maintenance algorithm processing these join subqueries in parallel by using referential integrity constraints over source relations. A relation which has several foreign keys can be joined with referenced relations independently. The proposed algorithm processes these join operations in parallel then it merges their results. With the parallel processing, the algorithm can maintain materialized views efficiently. We show the superiority of the proposed algorithm using an analytical cost model.

## 1 Introduction

Data Warehouses (DW) are composed of materialized views which maintain online analytical information extracted from data sources located physically at different sites. These materialized views can be used to process users' OLAP queries efficiently without accessing data sources [1,2,3]. Whenever data sources are changed, the materialized views defined on the sources should be maintained to keep them up-to-date. This process is called view maintenance. Of view maintenance methods, incremental maintenance has been widely used, because it incrementally maintains views by using only the updated portion of each source relation [4,5].

Data sources for data warehouses can be stored in remote areas and they can be changed independently. It is difficult to maintain views consistently over the



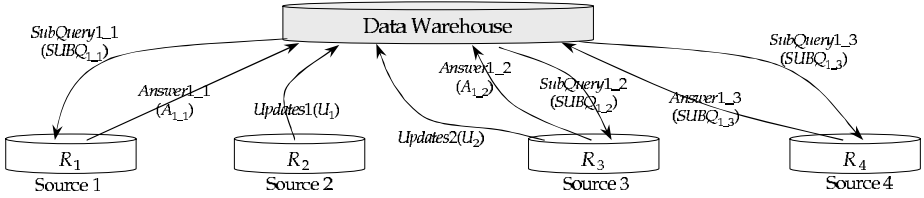
changes of data sources. Thus we need a technique which maintains views as the same sequence of changes and the same state as data sources. We call it view consistency maintenance [3,4]. There have been several algorithms for view consistency maintenance, such as ECA [5], Strobe [6], SWEEP [7], and PVM [8]. The ECA algorithm was developed for view maintenance in single source site. The Strobe and SWEEP algorithms proposed a view consistency maintenance for join views on source relations distributed at multiple sites. The PVM suggested a parallel processing algorithm for view maintenance, which handles multiple updates occurred in source relations concurrently. When any change (i.e., insertion or deletion) occurs in a source relation, all of these algorithms has to process a sequence of join operations serially which combine the update with the tuples of other source relations. From the results of these join operations, we can get the values to modify views. Because each source relation can be stored in different sites, the join operations take much processing time. In this paper, we develop a parallel view maintenance algorithm, called PSWEEP/RI (Parallel SWEEP with Referential Integrity) which executes these join operations in parallel.

Let's suppose that a source relation has several referential integrity constraints and a materialized view is defined as joining it with other source relations through foreign keys. The join operations are independent from each other thus they can be executed in parallel to build the view. By using this property of referential integrity constraints, this paper proposes a parallel processing algorithm, PSWEEP/RI, for view consistency maintenance, which processes join operations for view maintenance in parallel. This can reduce the processing time for view maintenance. Furthermore, this can also lessen the problem that view maintenance cost increases linearly when the number of source relations does. The remainder of this paper is organized as follows. Section 2 describes the related works and the motivation of this paper. Section 3 presents the basic concepts of the proposed PSWEEP/RI algorithm. Section 4 analyzes the performance evaluation then, finally, Section 5 concludes this paper.

## 2 Related Works and Motivation

Materialized views in data warehouse are defined as join views on source relations distributed over several sites. When one of source relations is changed, the corresponding materialized views should be maintained to accommodate the source updates. For this view maintenance, data warehouse has to execute join operations of the changed source relation and all other source relations used to define the views.

For example, suppose there is a data warehouse view defined on four source relations shown in Figure 1. When some changes  $U_1$  (i.e.,  $\Delta R_2$ ) happen at the source relation  $r_2$  and they are sent to data warehouse, the data warehouse has to execute the following view maintenance query Query1 to compute the information ( $\Delta V$ ) changing its view. In order to execute Query1, as shown in the Figure 1, the subqueries to join  $\delta R_2$  and other source relations ( $R_1$ ,  $R_3$ , and



**Fig. 1.** View maintenance process over distributed sources

$R_4$ ) are sent in turn to the source sites and are executed [1,4] as follows. (Here,  $Answer1_1$  is the result of  $SubQuery1_1$ ).

$$\begin{aligned}
 Query1 &= R_1 \times R_2 \times R_3 \times R_4 \\
 SubQuery1_1 &= R_1 \times R_2 \\
 SubQuery1_2 &= Answer1_1 \times R_3 \\
 &\dots \quad \dots \quad \dots
 \end{aligned}$$

These source relations are stored in different sites each other and they can be changed independently at the same time. Thus it is very difficult to keep the consistency between materialized views and source relations. For example, when the source relation  $r_3$  happens some changes  $U_2$  (i.e.,  $\Delta R_3$ ) before the  $Answer1_2$  is computed at the site, the answer of the  $SubQuery1_2$  can involve the state that the updates  $U_2$  had already occurred, which is different from the state of source relation  $r_1$ . Thus the views may not be maintained consistently. Therefore, we need view maintenance techniques to guarantee the consistency between data warehouse views and distributed source relations. For these techniques, ECA, Strobe, SWEEP, and PVM algorithms have been proposed [5,6,7,8].

Zhuge et al. [5] introduced ECA algorithms ensuring the consistency of views for the cases that source relations are stored at a single source site. The Strobe algorithm [6] considers distributed data sources, requiring materialized views to contain a key for each of base relations and also requiring quiescence before installing any changes in materialized views. The SWEEP algorithms are introduced by executing serially view maintenance queries for updates in source relations as the order of their arrivals [7]. In the SWEEP algorithm, the join operations included in each view maintenance query are executed sequentially like Figure 1. It can incur much processing time when many resource relations are involved and/or source relation updates happen very frequently. In order to solve the problems of the SWEEP, the PVM algorithm [8] extends the SWEEP by invoking and parallelizing multiple threads for view maintenance, one thread for a view maintenance query. However, each thread executes the join operations of its query sequentially as same as the SWEEP does. Therefore, this paper proposes another approach for view consistency maintenance executing the join subqueries in parallel. By doing this, we can process efficiently view maintenance queries and we can also reduce the problem that view maintenance cost increases as linear as the number of source relations increases.

### 3 PSWEEP/RI Algorithm

This chapter describes the basic concepts of PSWEEP/RI which we propose. The algorithm processes join operations among source relations in parallel or it filters out view maintenance operation without processing any join among them [1].

#### 3.1 Strategies in the SWEEP Algorithm

Let's suppose that there are referential integrity constraints on source relations and materialized views are defined as the joins among them. These referential integrity constraints can be represented by a graph as shown in the Figure 2. In the figure, ENROL relation refers to STUDENT, COURSE, and PROFESSOR relations. We call that it has *referring relationship*. When a relation refers to multiple relations, it is called *multiple referring*. Some relations such as STUDENT, COURSE, and PROFESSOR can be referred by others. We call they have *referred relationships*. When a relation is referred by several relations, it is called *multiple referred*.

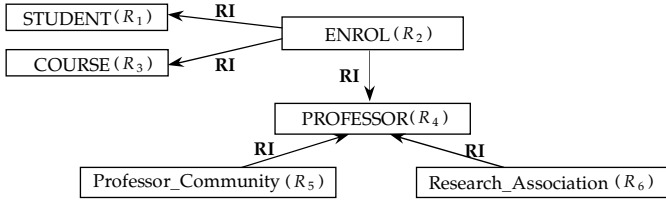


Fig. 2. Referential Integrity (RI) graph

When some changes(i.e.,  $\Delta R_2$ ) occurs in a relation  $R_2$  (i.e., ENTROL), in this example, data warehouse has to perform the following query to get the information for maintaining its views:

$$\Delta V = R_1 \bowtie \Delta R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5 \bowtie R_6$$

In the existing SWEEP algorithm, this query is executed as follows: the join operations in the left-hand side of  $\Delta R_2$  are processed at first then the rest join operations in the right side of  $\Delta R_2$  are done. (Refer to the following procedure.) These join operations are executed in sequential order. In order to perform each join operation, furthermore, a subquery is sent to the site storing the relation to be joined. If the number of source relations defining a view increases, the number of join operations to execute increases thus the processing time for view maintenance increases too.

$$\begin{aligned} \text{Left Sweep: } \Delta V_{\text{left}} &= R_1 \bowtie \Delta R_2 \\ \text{Right Sweep: } \Delta V &= \Delta V_{\text{left}} \bowtie R_3 \bowtie R_4 \bowtie R_5 \bowtie R_6 \end{aligned}$$

### 3.2 Maintenance Strategy for Multiple Referring Relation

Suppose some changes (i.e.,  $\Delta R_2$ ) occur at a multiple referring relation (i.e.,  $R_2$ ) as shown in the above example. From RI graph described above section, we can identify that the relation  $R_2$  refers to  $R_1$ ,  $R_3$ , and  $R_4$ . In order to get the view maintenance information( $\Delta V$ ), the join operations between  $\Delta R_2$  and  $R_1$ ,  $R_3$ , or  $R_4$  should be executed. To prove it, we present some lemmas and a theorem in the below. Here we assume that  $r_1$ ,  $r_2$ ,  $r_3$ , and  $r_4$  relations are defined the schema  $R_1$ ,  $R_2$ ,  $R_3$ , and  $R_4$  respectively, and  $r_1 \rightarrow r_2$  indicates that relation  $r_1$  refers to  $r_2$ .

**Lemma 1.** *Self-join of an identical relation  $r_1$  using its primary key, produces the same relation as the original one. That is,  $\pi_{R_1}(r_1 \bowtie r_1) = r_1$  holds.*

**Lemma 2.** *(commutative rule) [9] Join operation is commutative. That is,  $r_1 \bowtie r_2 = r_2 \bowtie r_1$  holds.*

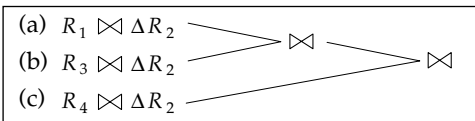
**Lemma 3.** *(associative rule) [9] Join operation is associative. That is,  $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$  holds.*

**Theorem 1.** *(parallel join using referential integrity) Let referential integrity constraints  $r_1 \rightarrow r_2$ ,  $r_1 \rightarrow r_3$ , and  $r_1 \rightarrow r_4$  exist and  $\Delta r_1$  be some changes of a referring relation  $r_1$ . Then  $\Delta r_1 \bowtie r_2 \bowtie r_3 \bowtie r_4 = \pi_{R_1, R_2, R_3, R_4}(\Delta r_1 \bowtie r_2) \bowtie (\Delta r_1 \bowtie r_3) \bowtie (\Delta r_1 \bowtie r_4)$  holds.*

PROOF: Let  $\Delta r_1$  be  $r_1$  for the simplicity, then the theorem is proved as follows.

$$\begin{aligned}
 & (r_1 \bowtie r_2) \bowtie (r_1 \bowtie r_3) \bowtie (r_1 \bowtie r_4) \\
 &= (r_1 \bowtie r_2 \bowtie r_1) \bowtie r_3 \bowtie (r_1 \bowtie r_4) && \text{by Lemma 3} \\
 &= (r_1 \bowtie r_1 \bowtie r_2) \bowtie r_3 \bowtie (r_1 \bowtie r_4) && \text{by Lemma 2} \\
 &= (\pi_{R_1}(r_1 \bowtie r_1) \bowtie r_2) \bowtie r_3 \bowtie (r_1 \bowtie r_4) \\
 &= (r_1 \bowtie r_2) \bowtie r_3 \bowtie (r_1 \bowtie r_4) && \text{by Lemma 1} \\
 &= (r_1 \bowtie r_2) \bowtie (r_1 \bowtie r_4) \bowtie r_3 && \text{by Lemma 2} \\
 &= (r_1 \bowtie r_2 \bowtie r_1) \bowtie r_4 \bowtie r_3 && \text{by Lemma 3} \\
 &= (r_1 \bowtie r_1 \bowtie r_2) \bowtie r_4 \bowtie r_3 && \text{by Lemma 2} \\
 &= (\pi_{R_1}(r_1 \bowtie r_1) \bowtie r_2) \bowtie r_4 \bowtie r_3 \\
 &= (r_1 \bowtie r_2) \bowtie r_4 \bowtie r_3 && \text{by Lemma 1} \\
 &= r_1 \bowtie r_2 \bowtie r_3 \bowtie r_4 && \text{by Lemma 2} \quad \square
 \end{aligned}$$

By the above Theorem 1, we can observe that the result of sequential execution for the join operations between  $\Delta R_1$  and  $R_2$ ,  $R_3$ , or  $R_4$  is equal to the one of parallel execution for them. Therefore, we process the view maintenance query in parallel as follows.



That is, the join subqueries (a), (b), and (c) are executed in parallel then their results are merged in turn to get the final result. (For the simplicity of explanation, merging was represented by a join operation.)

### 3.3 Maintenance Strategy for Referred Relation

This section describes the strategies for changes on a referred relation, which is referred by other relations.

**Theorem 2.** (*filtering equijoin using referential integrity*) Suppose that  $r_1 \rightarrow r_2$  exist and the changes on the referred relation  $r_2$  is  $\Delta r_2$ . Then  $r_1 \bowtie \Delta r_2 = \phi$ , where  $\Delta r_2$  is the combination of insertions (notated by  $\blacktriangle r_2$ ) and deletions (notated by  $\blacktriangledown r_2$ ).

PROOF:

- 1) When some insertions occurred at  $r_2$ , if  $r_1 \bowtie \blacktriangle r_2 \neq \phi$ , then it means that the referring relation  $r_1$  has tuples satisfying  $r_1.FK = \blacktriangle r_2.PK$ . These tuples violate the referential integrity constraint (i.e.,  $r_1 \rightarrow r_2$ ), it contradicts the definition of referential integrity.
  - 2) When some deletions occurred at  $r_2$ ,
    - 2.1) if there is any tuple in  $r_1$  that the deleted tuples in  $\blacktriangledown r_2$  refer to, it will be automatically deleted or changed into another value according to the options (i.e., cascade, set null, set default, or restrict) given to the referential integrity constraint. Therefore  $r_1$  doesn't have any tuple to join with  $\blacktriangledown r_2$  thus  $r_1 \bowtie \blacktriangledown r_2 = \phi$ . Otherwise,  $\blacktriangledown r_2$  couldn't be performed.
    - 2.2) if there are no tuples in  $r_1$  that refer to the tuples deleted from  $r_2$ , there are no tuples to join with  $\blacktriangledown r_2$ . Therefore  $r_1 \bowtie \blacktriangledown r_2 = \phi$  must be true.
- With both 1) and 2) above,  $r_1 \bowtie \Delta r_2 = \phi$  is true. □

As proved in the Theorem 2, even though any tuple is inserted into or deleted a referring relation, the result of join operations for view maintenance queries is always null. In the proposed PSWEEP/RI algorithm, the maintenance queries for the changes of referred relations are filtered without executing any operations because they have no affection to the view updates.

## 4 Performance Evaluation

In this chapter, we present the results of performance evaluation comparing the proposed PSWEEP/RI and the previous SWEEP. For the evaluation, we use the materialized view example in Figure 2. In Section 4.1, we present an analytical cost model based on the example in Figure 2. In Sections 4.2~4.4, we show the evaluation results computed by using the cost model.

### 4.1 PSWEEP/RI Cost Model

To design the analytical cost model for the example in Figure 2, we use the parameters described in Table 1. And, we also use the following assumptions to analyze the proposed algorithm.

- Since referential integrity constraints are enforced, the *index join* method will be used if a join has a referential integrity constraint.
- The *nested loop join* method will be used if a join has not any referential integrity constraint.

**Table 1.** Summary of parameters

Parameters	Definition/Meaning
$Ccom$	Bandwidth for transmitting data
$N(R_i)$	Number of tuples in relation $R_i$
$W(R_i)$	Average size of a tuple in relation $R_i$
$N(dR_i)$	Number of tuples in $R_i$ delta relation
$DW$	Data warehouse
$\Delta R_i$	Changed parts of the source relation $R_i$
$Send(\Delta R_i, DW)$	Cost for transmitting $\Delta R_i$ to $DW$
$Join(\Delta R_i, R_j)$	Cost for joining $\Delta R_i$ and $R_j$
$B$	Block size
$k$	Cost for depth first search in general case
$br$	Constant value for communication overhead

- Changes will be occurred in relation  $R_2$  of the example, and let the changed parts of relation  $R_2$  be  $\Delta R_2$ .

We compute the total view maintenance cost of PSWEEP/RI by adding communication cost and join cost. Since communication operations in the same phase are able to be processed in parallel, the cost for these operations will be set to the maximum cost of them. Also, since join operations in the same phase are able to be processed in parallel too, the cost for these join operations will be set the maximum cost of them. The following equations show the analytical cost model of PSWEEP/RI for the example in Figure 2 (The cost model of the existing algorithm SWEEP can be derived as the similar way, but it is omitted due to space limitation.).

$$\textcircled{1} \quad Send(\Delta R_2, DW) = (N(dR_2) \cdot W(R_2)) / Ccom \quad Eq. (1)$$

$$\textcircled{2} \quad Send(\Delta R_{2(DW)}, R_1) = (N(dR_2) \cdot W(R_2)) / Ccom \quad Eq. (2)$$

$$\textcircled{2} \quad Send(\Delta R_{2(DW)}, R_3) = (N(dR_2) \cdot W(R_2)) / Ccom \quad Eq. (3)$$

$$\textcircled{2} \quad Send(\Delta R_{2(DW)}, R_4) = (N(dR_2) \cdot W(R_2)) / Ccom \quad Eq. (4)$$

$$\textcircled{3} \quad Join(\Delta R_2, R_1) = N(dR_2) \cdot [\log_k N(R_1) + 1] \cdot br = X \quad Eq. (5)$$

$$\textcircled{3} \quad Join(\Delta R_2, R_3) = N(dR_2) \cdot [\log_k N(R_3) + 1] \cdot br = Y \quad Eq. (6)$$

$$\textcircled{3} \quad Join(\Delta R_2, R_4) = N(dR_2) \cdot [\log_k N(R_4) + 1] \cdot br = Z \quad Eq. (7)$$

$$\textcircled{4} \quad Send(X, DW) = ((N(dR_2) \cdot (W(R_2) + W(R_1))) \cdot 8) / Ccom \quad Eq. (8)$$

$$\textcircled{4} \quad Send(Y, DW) = ((N(dR_2) \cdot (W(R_2) + W(R_3))) \cdot 8) / Ccom \quad Eq. (9)$$

$$\textcircled{4} \quad Send(Z, DW) = ((N(dR_2) \cdot (W(R_2) + W(R_4))) \cdot 8) / Ccom \quad Eq. (10)$$

$$\textcircled{5} \quad Join(X, Y) = ([N(dR_2) \cdot (W(R_2) + W(R_1))]/B) + \\ [N(dR_2) \cdot (W(R_2) + W(R_1))]/B] \cdot br = M \quad Eq. (11)$$

$$\textcircled{6} \quad Join(M, Z) = ([N(dR_2) \cdot (W(R_2) + W(R_1) + W(R_3))]/B) + \\ [N(dR_2) \cdot (W(R_2) + W(R_4))]/B] \cdot br = N \quad Eq. (12)$$

$$\textcircled{7} \quad Send(N_{(DW)}, R_5) = (N(dR_2) \cdot (\sum_{i=1}^4 W(R_i)) \cdot 8) / Ccom \quad Eq. (13)$$

$$\textcircled{7} \quad Send(N_{(DW)}, R_6) = (N(dR_2) \cdot (\sum_{i=1}^4 W(R_i)) \cdot 8) / Ccom \quad Eq. (14)$$

$$\textcircled{8} \quad Join(N, R_5) = N(dR_2) \cdot \left( [\log_k N(R_5)] + \frac{N(R_5)}{N(R_4)} \right) \cdot br = I \quad Eq. (15)$$

$$\textcircled{8} \quad Join(N, R_6) = N(dR_2) \cdot \left( [\log_k N(R_6)] + \frac{N(R_6)}{N(R_4)} \right) \cdot br = J \quad Eq. (16)$$

$$\textcircled{9} \quad Send(I, DW) = \left( N(dR_2) \cdot \frac{N(R_5)}{N(R_4)} \cdot (\sum_{i=1}^4 W(R_i) + W(R_5)) \cdot 8 \right) / Ccom \quad Eq. (17)$$

$$\textcircled{9} \text{ Send}(J, DW) = \left( N(dR_2) \cdot \frac{N(R_6)}{N(R_4)} \cdot (\sum_{i=1}^4 W(R_i) + W(R_6)) \cdot 8 \right) / Ccom \quad \text{Eq. (18)}$$

$$\textcircled{10} \text{ Join}(I, J) = \left( \left[ N(dR_2) \cdot \frac{N(R_5)}{N(R_4)} \cdot (\sum_{i=1}^4 W(R_i) + W(R_5)) \right] / B \right) + \left[ \left( N(dR_2) \cdot \frac{N(R_6)}{N(R_4)} \cdot (\sum_{i=1}^4 W(R_i) + W(R_6)) \right) / B \right] \cdot br \quad \text{Eq. (19)}$$

Total communication cost = Eq.(1) + max{Eq.(2), Eq.(3), Eq.(4)} + max{Eq.(8), Eq.(9), Eq.(10)} + max{Eq.(13), Eq.(14)} + max{Eq.(17), Eq.(18)}

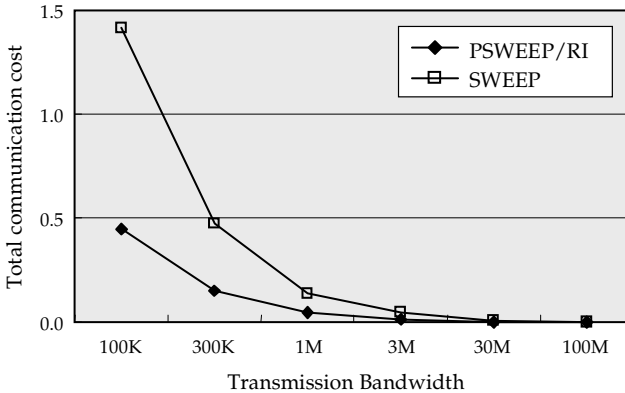
Total maintenance cost (join cost + communication cost) = Eq.(1) + max{Eq.(2), Eq.(3), Eq.(4)} + max{Eq.(5), Eq.(6), Eq.(7)} + max{Eq.(8), Eq.(9), Eq.(10)} + Eq.(11) + Eq.(12) + max{Eq.(13), Eq.(14)} + max{Eq.(15), Eq.(16)} + max{Eq.(17), Eq.(18)} + Eq.(19)

## 4.2 Experimental Data and Environment

In analytical experiments, we compute the cost required in PSWEEP/RI and that in SWEEP by varying transmission bandwidth, relation size, and the number of tuples. We determine the database sizes on the basis of TPC-D, and adjust the other parameter values based on TPC-D database sizes. Table 2 shows the parameter values used in the experiments.

**Table 2.** Parameter values used in the analytical experiments

Parameters	Values
$Ccom$	100Kbps ~ 100Mbps
$W(R_i)$	250 bytes
$B$	1,024 bytes
$k$	31
$br$	0.02



**Fig. 3.** Results of communication cost by varying the transmission bandwidth

### 4.3 Evaluation Results for Different Transmission Bandwidths

Figure 3 shows the changes of communication cost on different transmission bandwidths. The sizes of source relations have the same values determined in Section 4.2, and we set the number of updated tuples,  $N(dR_i)$ , to one and compute the costs by increasing the transmission bandwidth from 100Kbps to 100Mbps. As shown in Figure 3, we know that the communication cost of the proposed PSWEEP/RI is always less than that of the previous SWEEP. In particular, in the cases of lower bandwidths, PSWEEP/RI outperforms SWEEP significantly.

### 4.4 Evaluation Results for Different Relation Sizes

Figure 4 shows the changes of total communication cost on different numbers of updated tuples, i.e., on different  $N(dR_2)$ 's. We compute the cost by increasing  $N(dR_2)$  from 1 to 32. As shown in the figure, we know that the communication cost of the PSWEEP/RI is always less than that of SWEEP due to the effect

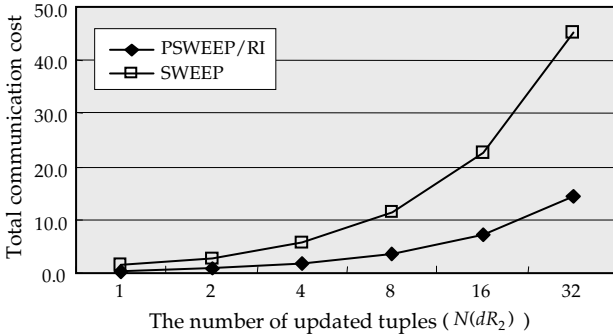


Fig. 4. Results of communication cost by varying the number of updated tuples

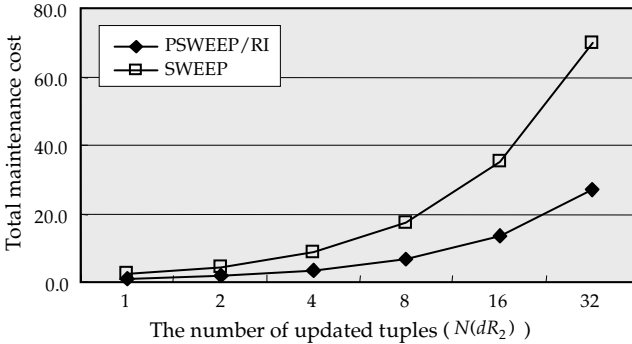


Fig. 5. Results of the total view maintenance cost by varying the number of updated tuples



of parallel processing. In particular, as  $N(dR_2)$  increases, the cost difference becomes larger.

Next, Figure 5 shows the total view maintenance costs of two methods when we fix the bandwidth to 1Mbps and increase the number of updated tuples,  $N(dR_2)$ , from 1 to 32. As shown in the figure, in the case where the number of updated tuples is 32, which is only 0.002% of the source relation, the proposed PSWEEP/RI improves performance by 159% over the previous SWEEP.

## 5 Conclusions

In this paper, we proposed a view maintenance method which guarantees the consistency between data warehouse views and distributed source data efficiently by using parallel processing techniques. Existing algorithms like SWEEP execute sequentially join operations on source relations which are invoked to get the data for view maintenance. This sequential execution requires high processing cost for view maintenance. In order to reduce the processing cost, we proposed an algorithm, PSWEEP/RI, processing join operations on source relations stored different sites in parallel. We also designed a cost model to measure the processing cost of the proposed algorithm and evaluated its performance. From the experiments, we found that the proposed algorithm reduced both the communication cost and the total processing cost compared to existing methods.

The basic approach of the proposed algorithm is to take advantage of referential integrity properties in order to parallelize join subqueries involved in view maintenance. The join operations between a source relation (including multiple foreign keys) and its referenced relations (stored at different site) can be executed independently (i.e., in parallel). Then the join results can be merged together to obtain the final result for view maintenance. Furthermore, any changes on each referenced relation does not have any tuple in referenced relations. Thus view maintenance queries invoked by referenced relations can be filtered out without executing any join operations [1]. Because of these properties, the proposed algorithm reduced the cost of processing a sequence of join operations and the communication cost. Because of parallel processing, the algorithm can also reduce the problem that the view maintenance cost of existing methods increases in proportional to the number of source relations. For further researches, the proposed algorithm can be extended for join operations of source relations which doesn't employ referential integrity. This algorithm considered only join operations hence we also need to investigate parallel processing techniques for complex views involving other algebraic operations and aggregate functions.

**Acknowledgements.** This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

## References

1. Quass, D., Gupta, A., Mumick, I. S., and Widom, J., "Making Views Self-Maintainable for Data Warehousing," In *Proc. of Conf. on PDIS*, pp.158-169, 1996.
2. Colby, L., et al., "Supporting Multiple View Maintenance Policies," In *Proc. of ACM SIGMOD Conf.*, pp.405-416, 1997.
3. Gupta, A. and Mumick, I. S., "Maintenance of Materialized Views: Problems, Techniques, and Applications," *IEEE Data Engineering Bulletin*, Special Issue on Materialized views and Warehousing, Vol. 18, No. 2, pp.3-18, 1995.
4. Ross, K. A., Stivastava, D., and Sudarshan, S., "Materialized View Maintenance and Integrity Constraint Checking: Trading Space for Time," In *Proc. of Proc. of ACM SIGMOD Conf.*, pp.447-458, 1996.
5. Zhuge, Y., et al., "View Maintenance in a Warehousing Environment," In *Proc. of Proc. of ACM SIGMOD Conf.*, pp.316-327, 1995.
6. Zhuge, Y., Garcia-Molina, H., and Wiener, J. L., "The Strobe Algorithms for Multi-Source Warehouse Consistency," In *Proc. of Conf. on PDIS*, pp.146-157, 1996.
7. Agrawal, D., El Abbadi, A., Singh, A., and Yurek, T., "Efficient View Maintenance at Data Warehouses," In *Proc. of Proc. of ACM SIGMOD Conf.*, pp.417-427, 1997.
8. Zhang, X., Ding, L., and Rundensteiner, E. A., "Parallel Multisource View Maintenance," *The VLDB Journal*, Vol. 13, No. 1, pp.22-48, 2004.
9. Navathe, S. and Elmasri, R. *Fundamentals of Database Systems*, 4th ed., Addison Wesley, 2004.

# Selective View Materialization in a Spatial Data Warehouse\*

Songmei Yu, Vijayalakshmi Atluri, and Nabil Adam

MSIS Department and CIMIC,  
Rutgers University, NJ, USA  
{songmei, atluri, adam}@cimic.rutgers.edu

**Abstract.** A spatial data warehouse (SDW) consists of a set of materialized views defined over the source relations, either conventional, spatial, or both. Often, when compared to the traditional data warehouses, the cost of view materialization is more expensive with respect to both computation and space. This is because the spatial data is typically larger in size, which leads to high maintenance cost, and the spatial operations are more expensive to process. In this paper, we address the issue of optimizing the view materialization cost in an SDW. We build a cost model to measure the on-the-fly computation cost versus the space cost for spatial queries. We show that a spatial query can be represented in the form of the query-graph and propose three transformation rules, edge-elimination, query-splitting and query-joining, to selectively materialize spatial views. We present a greedy algorithm for materialized view selection so that the local cost optimality can be achieved.

## 1 Introduction

With the popular use of satellite telemetry systems, remote sensing systems, medical imaging, and other computerized data collection tools, a huge amount of spatial data is increasingly being stored and managed in spatial information repositories. A spatial data warehouse (SDW) [1] is a subject-oriented, integrated, time-variant, and non-volatile collection of both spatial and non-spatial data in support of decision-making processes.

An SDW consists of a set of materialized views defined over the source relations, either conventional, spatial, or both. If an input query can be evaluated by the materialized views, then this is performed through partial or complete rewriting of the input queries over the materialized views. As such, storing views incurs space cost. Moreover, when the source relations change, the corresponding materialized views need to be updated, which incurs the view maintenance cost. Although storage is not a big issue given the lower hardware cost, the view maintenance cost is normally increased as the size of views gets larger [2]. On the other hand, if an input query cannot be rewritten over the materialized views, then it

---

\* The work is supported in part by the New Jersey Meadowlands Commission under the project Meadowlands Environmental Research Institute.

has to be computed on-the-fly and hence incurs the online computational cost that effects the query response time. Therefore, selectively materializing views has become a philosophy in designing a data warehouse, which could enhance the query performance through query rewriting and help achieving the desired cost optimization.

This problem has been well studied in the traditional warehouse for SPJ (selection, project and join) queries[2,3,4]. However, in an SDW, the cost of view materialization is more expensive with respect to both computation and space. This is because, spatial data is typically larger in size, and the spatial operations, such as region merging, map overlaying, and spatial range selection, are more expensive to process. If one resorts to materialize all spatial queries to improve response time, their space cost and the maintenance cost may significantly increase. In this paper, we propose (1) a *cost model* to balance the spatial query response time and space cost, and (2) a set of *transformation rules* that aid in the process of selective materialization of spatial views for cost minimization.

There are two basic ways to represent queries in graphs. The first is using an AND/OR DAG (directed acyclic graph) to represent an algebraic expression, where nodes in the DAG are partitioned into AND nodes for relational operators and OR nodes for relational expressions, and the edges represent the directions or sequences of query operations [2,3]. However, using DAG to detect common sub-expressions is an expensive process since it detects all common sub-expressions, and not all of them need to be materialized from the optimization point of view. The second method is based on the query definition. A single-query graph and a multi-query graph are introduced in [4] where the nodes are labeled by relations and edges are labeled by relational operations. A set of transformation rules is applied and all significant common sub-expressions are detected and materialized, which minimizes the cost model of a data warehouse. This technique is limited to SPJ queries on traditional relations and not able to handle spatial operations on spatial objects. Hence we want to explore the techniques to spatial domains, by considering not only the SPJ operations on alpha-numeric data, but also spatial operations on spatial attributes. The preliminary concepts of the SDW model and associated queries are presented in [5], and additional challenges involved in processing the spatial queries have identified in [6]. To the best of our knowledge, our work is the first to address the issue of optimizing the view materialization cost in an SDW.

In section 2, we present the preliminaries on hybrid algebra expressions for specifying spatial queries. In section 3, we present our cost model of an SDW. In section 4, we propose transformation rules for spatial query graphs, and present a greedy algorithm for materialized view selection so that the local cost optimality can be achieved. The paper is concluded in section 5.

## 2 Preliminaries

In this section, we briefly review the specification of the spatial queries using *hybrid algebra expressions*, which can be specified on *hybrid relations* using *hybrid algebra operators*[7].

A base relation of an SDW is a hybrid relation, which includes attributes and tuples from both alphanumeric relations and spatial relations. For spatial relations, we adopt the definitions from the standard specifications of Open Geospatial Consortium (OGC) {[www.opengeospatial.org](http://www.opengeospatial.org)}. The spatial data types supported by this standard are OGC Geometry Object Model (GOM), where the geometry class serves as the base class with sub-classes for *Point*, *Line* and *Polygon*, as well as a parallel class of *geometry collection* designed to handle a collection of points, lines and polygons. Conceptually, spatial entities are stored as relations with geometry valued attributes as columns, and their instances as rows. A *hybrid algebra operand* is a distinct attribute of a hybrid relation.

The set of *hybrid algebra operators* HO include:

1. Relational operators: RO =  $\{\sigma, \pi, \cup, -, \times\}$
2. Comparison operators: CO =  $\{=, <, \leq, >, \geq, \neq\}$
3. Aggregate operators [8]: AO =  $\{\text{distributive functions} \cup \text{algebraic functions} \cup \text{holistic functions}\}$ , where
  - distributive functions =  $\{\text{count, sum, min, max}\}$ <sup>1</sup>
  - algebraic functions =  $\{\text{avg, min}_N, \text{max}_N, \text{standard\_deviation}\}$   
such that N is a bounded positive integer
  - holistic functions =  $\{\text{median, rank}\}$   
such that no constant bound on the storage size needed to describe a sub-aggregate
4. Spatial operators: SO =  $\{\text{Spatial Basic Operators} \cup \text{Spatial Topological Operators} \cup \text{Spatial Analysis Operators}\}$ <sup>2</sup>
  - Spatial Basic Operators =  $\{\text{area, envelope, export, isEmpty, isSimple, boundary}\}$
  - Spatial Topological Operators =  $\{\text{equal, disjoint, intersect, touch, cross, within, contains, overlap, relate}\}$
  - Spatial Analysis Operators =  $\{\text{distance, buffer, convexHull, intersection, union, difference, symDifference}\}$

An atomic formula,  $f$ , can be either unary operation  $op(X_1)$ , binary operation  $op(X_1, X_2)$ , or n-nary operation  $op(X_1, \dots, X_n)$ , where  $op$  is a hybrid algebra operator and each  $X_i$  is a hybrid operand. Examples of atomic formulas are:  $\text{boundary}(\text{administrative\_map})$ ,  $\text{overlap}(\text{population\_map}, \text{administrative\_map})$ ,  $\text{intersection}(\text{map1}, \text{map2}, \text{map3})$ .

**Definition 1.** *Hybrid Algebra Expression:* A hybrid algebra expression is a formula  $F$ , which is defined as: (i)  $f$  is a hybrid algebra expression  $F$ , (ii) if  $F_1$  is a hybrid algebra expression, then  $F = op(X_1, \dots, X_m, F_1)$  is a hybrid algebra expression, and (iii) if  $F_1$  and  $F_2$  are two hybrid algebra expressions, then  $F_1 \wedge F_2$ ,  $F_1 \vee F_2$ ,  $\neg F_1$  and  $(F_1)$  are hybrid algebra expressions.

<sup>1</sup> Note that the distributive, algebraic and holistic functions are not exhaustive lists.

<sup>2</sup> We use well-defined spatial operators from the OGC standard.

### 3 Cost Model for a Spatial Data Warehouse

A spatial query typically consists of several atomic spatial operations, which are either spatial selections or spatial joins. If we decompose the query into component queries, the component query with only one single spatial operation is denoted as a simple spatial query (SSQ). SSQ essentially is nothing but an atomic formula which serves as the smallest unit for the purpose of cost measurement. From now on, we use  $q$  to denote an SSQ and  $p$  to denote a spatial query composed of several  $q$ , and each  $q$  is an un-divisible unit within  $p$ .

The following three factors should be considered to decide whether a  $q$  needs to be computed on-the-fly or be materialized:

1. *The On-the-fly Computation Cost:* We measure the on-the-fly computation cost of a  $q$ , denoted as  $P(q)$ , in terms of the query response time. Most of spatial query processing methods fall into two categories: spatial join and spatial selection. Theodoridis et al. in [9] have presented analytical models to estimate the response time for both selection and join queries using R-trees. In this paper, we adopt the unified cost model proposed by [9] to estimate the value of  $P(q)$ .
2. *The Potential Access Frequency:* Based on the access history one may estimate the access frequency of  $q$ , which is denoted as  $fr(q)$ , over a certain time period.
3. *Size:* Basically, an image/map size is measured in *Bytes* or *Megabytes* and determined by several factors, such as resolution (the number of pixels per inch) and channel (Bytes per pixel). The equation is as follows: Height (Pixels)  $\times$  Width (Pixels)  $\times$  Bytes per pixel (the number of channels)/1,048,576 = size in MB. We assume the size of the generated view from  $q$ , denoted as  $S(q)$ , is a fixed value at the data warehouse design level, by specifying the height, width, resolution and channels. Given the maximum space capacity of the SDW is  $L$ , obviously one must materialize a view only if  $S(q) \leq L$ .

Now that we have  $P(q)$ ,  $fr(q)$  and  $S(q)$ , we define the following formulas to measure *execution cost* of  $q$ ,  $p$  and the *total cost* of an SDW as follows:

$$C(q) = (P(q) \times fr(q)) / (S(q) / L) \quad (1)$$

$$C(p) = P(p_o) + \lambda(p_m) \quad (2)$$

$$C(Q, V) = \sum_{i=1}^n C(p_i) \quad (3)$$

In formula (1), for each  $q$  in an SDW, the more frequently or expensively to compute it or the less space its result takes, the larger the value of  $C(q)$  would be, and the more likely we need to materialize it. Therefore, we need a control threshold, which is denoted as  $\delta$  and used to control the cost of executing a  $q$ . The general rule is that if  $C(q)$  is greater than  $\delta$ , we materialize  $q$  otherwise we compute it on-the-fly. For a data warehouse with heavy spatial operations involved or with large space permit, we normally choose a lower control threshold to materialize more views for quick query response time.

By controlling the execution cost of each  $q$  within a spatial query  $p$ , we divide  $p$  into two subsets, one is denoted as  $p_o$ , for materialization and the other is denoted as  $p_m$ , for on-the-fly computation. Then we use  $p_o$  and  $p_m$  to rewrite  $p$ . Since the on-the-fly computational cost is the major part of the spatial query evaluation cost, and the maintenance cost is analogous to the space of materialized views [2], we compute the total cost of a spatial query  $p$  in formula (2). Here  $\lambda$  is a parameter between 0 and 1 indicating the relative importance of the spatial query computational cost vs. the space cost.

Obviously, the on-the-fly computation cost is monotonically decreasing as the number of materialized views increases, and the space cost is monotonically increased as more views are materialized. Our goal is to achieve an optimal number of materialized views to balance the on-the-fly computation cost and space cost thus the total cost of each spatial query is minimized. Given the set of queries  $Q = \{p_1, \dots, p_n\}$  and a set of materialized views  $V$  from  $Q$ , the total cost of an SDW is in formula (3).

## 4 Transformation Rules for Query Rewriting in an SDW

In this section, we first show that queries can be represented in a graphical form, which decompose a spatial query into several SSQs where each SSQ can be represented by an edge. We propose transformation rules, including edge transformation and query transformation, which aid in query rewriting during the process of deciding spatial view materialization. Essentially, we extend the transformation rules proposed for relational algebra expressions (limited to SPJ operations [4]) to hybrid algebra expressions. We finally present a greedy algorithm to guarantee that the cost incurred in answering a spatial query is minimal.

### 4.1 Graph Representation of Single and Multiple Queries

A spatial query defined on a hybrid algebra expression can be represented by a single-query graph as follows:

**Definition 2.** *Single Query Graph: Given a spatial query  $p$  specified over a set of hybrid relations  $R_1, \dots, R_m$ , a singlequery graph  $SG^p$  is a labeled graph where:*

1. *The nodes of  $SG^p$  is a set  $\{R_1, \dots, R_m\}$  on which  $p$  is defined. The node label  $NL_i$  for every node  $R_i$  is  $p : X, i \in [1, m]$  where  $X$  is the set of attributes of  $R_i$  projected out in  $p$ .*
2. *For every atomic formula  $q$  in  $p$* 
  - *if  $q$  is a unary operation on  $R_i$ , there is an edge from  $R_i$  to itself (self-operation loop) in  $SG^p$  with edge label  $EL_i$  as  $p : q$ .*
  - *if  $q$  is a binary operator on two relations  $R_i$  and  $R_j (i \neq j)$ , there is an edge between  $R_i$  and  $R_j$  (joining-operation edge) in  $SG^p$  with edge label  $EL_{ij}$  as  $p : q$ .*

Given a set of queries  $Q$ , the single-query graphs can be combined into a multi-query graph [4], which allows the compact representation of all the queries in one

graph so that it can be used in the view materialization and the query rewriting process.

**Definition 3.** *Multi-query Graph:* Given a set  $Q = \{p_1, \dots, p_n\}$  of queries, the corresponding multi-graph  $MG^Q$  is a labeled graph obtained by merging all single-query graphs  $\{SG^{p_1}, \dots, SG^{p_n}\}$  of the queries in  $Q$ , where:

1. The nodes of  $MG^Q = \cup_{i=1}^n \{\text{nodes of } SG^{p_i}\}$ .
2. If there exists a same node  $R_k$  in both  $SG^{p_i}$  and  $SG^{p_j}$  with node labels  $NL_{k_i}$  and  $NL_{k_j}$ , then there is only one node  $R_k$  in  $MG^Q$  with the node label as  $(NL_{k_i}, NL_{k_j})$ .
3. If there exists an edge between  $R_i$  and  $R_j$  ( $i$  and  $j$  not necessarily different) with edge label  $EL_{i_j}$  in  $SG^{p_k}$  then there exists the same edge with the same label in  $MG^Q$ .

*Example 1.* Consider two queries  $p_1$  and  $p_2$  defined over hybrid relations  $R_1(A, B, C)$ ,  $R_2(D, E)$ ,  $R_3(F, G)$ ,  $R_4(H, I, J)$ , in which  $A, D, F, H, J$  are spatial attributes, whose hybrid algebra expressions and corresponding  $SG^{p_1}$ ,  $SG^{p_2}$ , and  $MG^{(p_1, p_2)}$  are as follows:

$$p_1 = \pi_{ADE}(\sigma_{\text{equal}(A,F)}(R_1 \bowtie_{\text{intersect}(A,D)} (\sigma_{E < 2} R_2) \bowtie_{E > G} (R_3)))$$

$$p_2 = \pi_{BF GH}(\sigma_{B > I}(R_1 \bowtie_{\text{intersect}(A,D)} R_2 \bowtie_{\text{contain}(D,J)} (\sigma_{\text{isEmpty}(H)} R_4)))$$

## 4.2 Transformation Rules on a Multi-query Graph

The purpose of the transformations on a given spatial query  $p$  is to split all SSQs in  $p$  into two subsets,  $p_o$  and  $p_m$ , as we introduced in formula (2), where  $p_o$  is a set for on-the-fly computation and  $p_m$  is the set for materialization. Then  $p$  is answered by rewriting it over  $p_o$  and  $p_m$  correspondingly. In the following, we present three transformation rules as well as the query rewriting after the transformation.

**{Rule 1: edge-elimination.}** Let  $p$  be a spatial query in  $MG^Q$ , and  $e$  be an edge between nodes  $R_i$  and  $R_j$  ( $i$  and  $j$  are not necessarily different) in  $MG^Q$  labeled as  $p : q$ . Let the cost of executing  $q$  be  $C(q)$ .

1. If there exists another edge in  $MG^Q$  with label  $(p' : q)$  between  $R_i$  and  $R_j$ , then remove  $e$  from  $MG^Q$ , and replace all other occurrences of  $(p : q_i)$  in edge labels by  $(p_o : q_i)$ . Rewrite query  $p$  as  $p = \sigma_q(p_o)$ .
2. Otherwise, if  $C(q) \geq \delta$ , remove  $e$  from  $MG^Q$ , insert  $(p : q)$  into  $(p_m)$ , and replace all other occurrences of  $(p : q_i)$  in edge labels by  $(p_o : q_i)$ .
  - If there exists a path between  $R_i$  and  $R_j$  in  $MG^Q$ . Rewrite query  $p$  as  $p = \sigma_q(p_o)$ .
  - Otherwise, rewrite query  $p$  as  $p = \bowtie_q(p_o)$ .
3. Otherwise, i.e., if  $C(q) \leq \delta$ , replace all occurrences of  $(p : q_i)$  in edge labels by  $(p_o : q_i)$ . Rewrite query  $p$  as  $p = p_o$ .



By comparing the execution cost of an SSQ with the cost threshold, we decide whether an edge is removed for materialization or is kept for on-the-fly computation. The new query  $p_o$  or  $p_m$  are parts of  $p$  that need computation on-the-fly or materialization respectively, and are used to rewrite  $p$ . If the spatial query  $p$  is computed totally on-the-fly at the beginning, the space cost is zero,  $C(p) = P(p)$ . After the edge-transformations, the on-the-fly computation cost may be decreased at the expense of increased space cost, i.e.,  $C(p) = P(p_o) + \lambda S(p_m)$ . This leads us towards the optimal total cost condition of  $C(p)$ .

The purpose of performing the query transformation is to identify minimum cost query, by either creating two new queries through breaking it where an existing query can be rewritten using exclusively the new queries, or by creating one new query through merging two queries where the two existing queries can be rewritten exclusively on the new query. A query threshold,  $\theta$ , is used to decide if the query can be split or two queries can be joined based on their respective computational cost and the space cost.

**{Rule 2: query-splitting.}** Let  $p$  be a spatial query in  $MG^Q$ , and  $N_1$  and  $N_2$  be two set of nodes labeled by  $p$ , such that (1)  $N_1 \cap N_2 = \phi$ , (2)  $N_1 \cup N_2$  is the set of all nodes labeled by  $p$ , and (3)  $C(p) > \theta$ . We want to split a query that has the cost greater than the query threshold. Assume  $E$  is the set of all edges connecting nodes in  $N_1$  and nodes in  $N_2$  labeled by a set  $(p : q_1, \dots, p : q_k)$ . Assume the on-the-fly computation cost and frequency of each  $q_i$  are  $C(q_i)$  and  $fr(q_i)$ . We compute the cost of executing  $p : q_i$ , denoted as  $C(E) = \sum_{(i=1)}^n ((C(q_i) \times fr(q_i)) / fr(q_i))$ . This is the weighted average cost of all atomic operations connecting nodes in  $N_1$  and nodes in  $N_2$ .

1. Replace every occurrence of  $p$  in  $N_1$  and  $N_2$  by  $p_1$  and  $p_2$  respectively.
2. If  $C(E) \geq \theta$ , remove all edges in  $E$  from  $MG^Q$ , and insert  $(p : q_1, \dots, p : q_k)$  into  $p_m$ , and replace all other occurrences of  $p$  in edge labels by  $p_o$ , and  $p = \sigma_{q_1, \dots, q_{i-1}, q_{i+1}, \dots, q_k} (p_1 \bowtie_{q_i} p_2)$  where  $q_i \in (q_1, \dots, q_k)$ .
3. Otherwise, replace every occurrence of  $p$  on  $E$  as either  $p_1$  or  $p_2$ , and  $p = (p_1 \bowtie_{q_i} p_2)$  where  $q_i \in (q_1, \dots, q_k)$ .

By performing the query-splitting, we break an expensive query into two smaller parts. Furthermore, by identifying the spatial operations between them for either materialization or on-the-fly computation, we are searching an optimal point between on-the-fly computation cost and space cost.

**{Rule 3: query-joining.}** Essentially, the purpose of this query-joining operation is to identify the queries whose total cost is smaller than the query threshold that have overlapping nodes and perform equivalent operations, and merge them into a new query. Let  $p_1$  and  $p_2$  be two spatial queries in  $MG^Q$ , and  $N_1$  and  $N_2$  be the sets of nodes labeled by  $p_1$  and  $p_2$  respectively such that (1)  $N_1 \cap N_2 \neq \phi$ , (2) there exists at least one edge  $e$  among nodes in  $N_1$  with label  $(p_1 : q)$  and another edge  $e'$  among nodes in  $N_2$  with label  $(p_2 : q)$ , and (3)  $C(p_1) + C(p_2) \leq \theta$ .

1. Remove  $e$ , replace all other occurrences of  $(p_1 : q_{1i})$  in edge labels by  $(p : q_{1i})$  and all other occurrences of  $(p_2 : q_{2i})$  in edge labels by  $(p : q_{2i})$ .

2. For every node in  $((N_1 \cup N_2) - (N_1 \cap N_2))$ , replace the node label  $(p_1 : X_1)$  and  $(p_2 : X_2)$  with  $(p : X_1)$  and  $(p : X_2)$  respectively. For every node in  $N_1 \cap N_2$ , replace the node label  $(p_1 : X_1, p_2 : X_2)$  with  $(p : (X_1 \cup X_2))$ .
3. Rewrite query  $p_1$  and  $p_2$  as  $p_1 = \pi_{X_1} \sigma_{(q_{11}, \dots, q_{1n})} p$ ,  $p_2 = \pi_{X_2} \sigma_{(q_{21}, \dots, q_{2m})} p$ .

By performing the query-joining, we merge two smaller views into a bigger one. It can be recursively executed in order to achieve an optimal view for further query processing with desirable cost value.

*Example 2.* We want to apply three transformation rules to figure (c) separately as follows: (1) remove the edge between node  $R_1$  and node  $R_2$ , (2) split  $p_1$  into two views  $p_{11}$  and  $p_{12}$ , and (3) join  $p_1$  and  $p_2$  into a new query  $p$ . The result is in figure (a), (b) and (c) respectively.

In figure 1(c), there are two identical edges between  $R_1$  and  $R_2$ , so we remove the one labeled by  $p_1$  and replace  $p_1$  as  $p_{1o}$ . Assume the cost of  $(p_2 : \text{contain}(A, D))$  is greater than  $\delta$ , then we remove  $(p_2 : \text{contain}(A, D))$  and insert it into  $p_{2m}$ , and replace  $p_2$  in other edge labels as  $p_{2o}$ . The result is in figure 2(a). In figure 1(c),

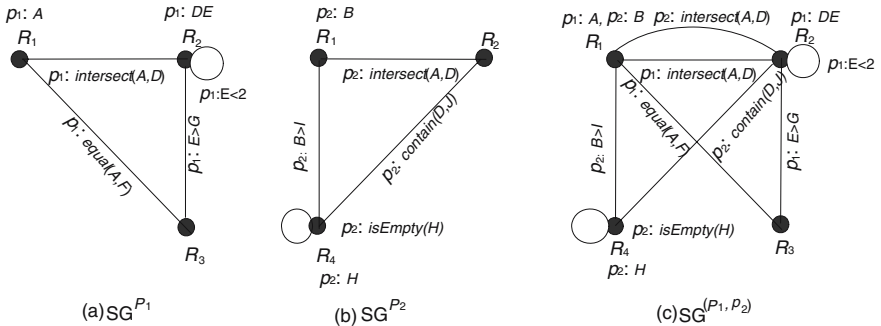


Fig. 1. Sample single-query graphs and multi-query graph

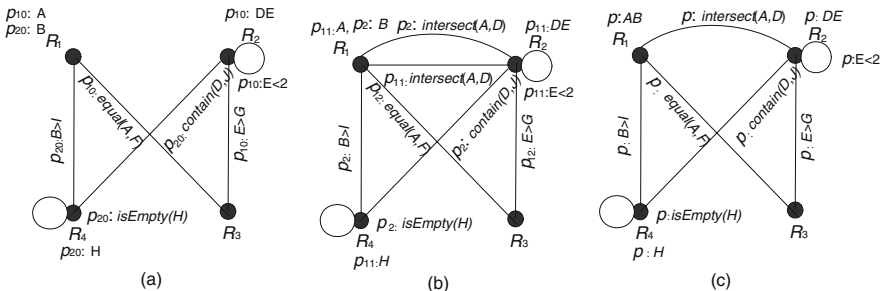


Fig. 2. A multi-query graph  $MG^Q$  after application of transformation rules

we split  $p_1$  into two views  $p_{11}$  and  $p_{12}$  if the cost of  $p_1$  is greater than  $\theta$ . The set of nodes labeled by  $p_1$  is  $\{R_1, R_2, R_3\}$ . We choose  $N_1$  as  $\{R_1, R_2\}$  and  $N_2$  as  $\{R_3\}$ . After evaluating the cost of operations between  $N_1$  and  $N_2$ , we keep the edges between  $p_{11}$  and  $p_{12}$  and perform on-the-fly computation on them. The result is in figure 2 (b). In figure 1(c), we join query  $p_1$  and  $p_2$  into a new query  $p$  if the total cost of  $p_1$  and  $p_2$  is less than  $\theta$ .  $N_1 = \{R_1, R_2, R_3\}$ ,  $N_2 = \{R_1, R_2, R_4\}$ ,  $N_1 \cap N_2 = \{R_1, R_2\}$ . The result is in figure 2(c).

**Theorem 1:** Our transformation rules are sound and complete.<sup>3</sup>

### 4.3 An Algorithm for Selective Materialization of Spatial Views

We first define a *benefit function* to monitor the cost savings for an SDW. Let  $A$  be an arbitrary set of SSQs in a multi-graph  $MG^Q$ . The benefit of  $A$  with respect to  $V$ , an already selected set of materialized views, is denoted as  $B(A, V)$ , is defined as: (1) If  $C(Q, V \cup A) < C(Q, V)$ , then  $B(A, V) = C(Q, V) - C(Q, V \cup A)$ , otherwise  $B(A, V) = 0$ . (2)  $B(A, \phi)$  is called the absolute benefit of  $A$ . Specifically, for each set of SSQs chosen to materialize, we compare the total cost of the SDW with the one before the materialization. If the current cost is less than before, then the difference represents the benefit of selecting  $A$  for materialization, otherwise, there is no benefit.

---

**Algorithm 1** The spatial greedy algorithm for view materialization

---

**Require:**  $Q, P(q), fr(q), S(q)$  for each  $q, L, \delta, \theta$  and  $\lambda$ .

- 1: Draw a multi-query graph  $MG_0^Q, V = \phi$ .
  - 2: **for** each edge  $e$  in  $MG_0^Q$  **do**
  - 3:   **if**  $B(A, V)$  is maximized and  $S(V) \leq L$  **then**
  - 4:     do edge-elimination( $e$ ),  $V = V \cup A$
  - 5:   **end if**
  - 6: **end for**
  - 7: Get a multi-query graph  $MG_1^Q$ .
  - 8: **for** each query  $p$  in  $MG_1^Q$  **do**
  - 9:   **if**  $B(A, V)$  is maximized and  $S(V) \leq L$  **then**
  - 10:     do query-splitting( $p$ ),  $V = V \cup A$
  - 11:   **end if**
  - 12: **end for**
  - 13: Get a multi-query graph  $MG_2^Q$ .
  - 14: **for** each applicable pair of views  $p_1$  and  $p_2$  **do**
  - 15:   **if**  $B(A, V)$  is maximized and  $S(V) \leq L$  **then**
  - 16:     do query-joining( $p_1, p_2$ ),  $V = V \cup A$
  - 17:   **end if**
  - 18: **end for**
  - 19: Materialize all views in  $V$
- 

Now we present the spatial greedy algorithm that transforms input queries of a given SDW and select a set of SSQs as materialized views. In the algorithm

<sup>3</sup> The detailed proof can be found in [10].

1, we first compute the cost for each spatial expression associated with each edge in  $MG_0^Q$ , and does edge-elimination accordingly to maximize the benefit function. Then we compute the cost for each query  $p$  in  $MG_1^Q$ , and does query-splitting for each applicable query to maximize the benefit function. Then we examine the views in  $MG_2^Q$  and performs query-joining to maximize the benefit function. The final multi-query graph is generated to materialize all SSQs in  $V$ . This spatial greedy algorithm is a cost driven technique deriving a sub-optimal solution for a given SDW because it commits to a local maximum cost benefit at each iteration, although not every locally maximum choice can guarantee the global maximality.

The main objective of the algorithm 1 is to optimize the space-time tradeoff when developing views for materialization. This optimization problem is NP-complete, which is a straightforward reduction from Set-Cover[1]. Thus, we are motivated to look at heuristics to produce approximated solutions. The obvious choice of heuristic is a greedy algorithm, where we select a set of views for materialization that demonstrate to be the best choice based on what have been given so far. This approach is always fairly close to optimal and in some cases can be shown to produce the best possible selection of materialized views.

## 5 Conclusions

View materialization is an essential query optimization strategy for decision-support applications. In this paper, we have investigated the problem of selectively materializing views given input queries from an SDW, and guarantee the rewriting of the input queries over these materialized views will minimize the proposed cost function. A cost model is developed for measuring spatial queries, which considers measuring query computation cost, frequency and space cost. A set of transformation rules is introduced to rewrite spatial queries, which divides a spatial query into materialization part and on-the-fly computation part. A spatial greedy algorithm is finally introduced by using those transformation rules to achieve the local cost benefit maximization. We are currently working on the utilization of spatial metadata for materialization view selection.

## References

1. Stefanovic, N., Jan, J., Koperski, K.: Object-based selective materialization for efficient implementation of spatial data cubes. *IEEE Transactions on Knowledge and Data Engineering(TKDE)* **12** (2000) 938–958
2. Gupta, H., Mumick, I.: Selection of views to materialize in a data warehouse. *Transactions of Knowledge and Data Engineering (TKDE)* **17** (2005) 24–43
3. Theodoratos, D., Sellis, T.: Dynamic data warehouse design. In Mohania, M.K., Tjoa, A.M., eds.: *Proc. of Data Warehousing and Knowledge Discovery*. Volume 1676 of *Lecture Notes in Computer Science.*, Springer (1999) 1–10
4. Theodoratos, D., Ligoudistianos, S., Sellis, T.: View selection for designing the global data warehouse. *Data and Knowledge Engineering(DKE)* **39** (2001) 219–240

5. Adam, N., Atluri, V., Yu, S., Yesha, Y.: Efficient storage and management of environmental information. In Kobler, B., Hariharan, P., eds.: Proc. of the 19th IEEE Symposium on Mass Storage Systems, NASA (2002) 165–181
6. Adam, N., Atluri, V., Guo, D., Yu, S.: Chapter 18: Challenges in Environmental Data Warehousing and Mining. In: Data Mining: Next Generation Challenges and Future Directions. 1 edn. AAAI Press (2004) 315–335
7. Atluri, V., Yu, S., Adam, N.: Cascaded star model and cascaded olap for spatio-temporal data warehouses. submitted for publication (2005)
8. Shekhar, S., Lu, C., Tan, X., Chawla, S.: Map Cube: A Visualization Tool for Spatial Data Warehouses. In: Geographic Data Mining and Knowledge Discovery. 1 edn. Taylor and Francis (2001) 74–110
9. Theodoridis, Y., Stefanakis, E., Sellis, T.: Efficient cost model for spatial queries using r-trees. IEEE Transactions on Knowledge and Data Engineering(TKDE) **12** (2000) 19–32
10. Yu, S., Atluri, V., Adam, N.: Towards view materialization in a spatial data warehouse. Technical Report (2005)

# PMC: Select Materialized Cells in Data Cubes

Hongsong Li<sup>1</sup>, Houkuan Huang<sup>1</sup>, and Shijin Liu<sup>2</sup>

<sup>1</sup> School of Computer and Information Technology,  
Beijing Jiaotong University, Beijing, China, 100044  
mlhs@163.com, hkhuang@center.njtu.edu.cn

<sup>2</sup> Ticket Center, Passenger Traffic Department,  
Urumchi Railroad Bureau, Urumchi, China, 830011  
LiukingLiuking@21cn.com

**Abstract.** QC-Trees is one of the most storage-efficient structures for data cubes in a MOLAP system. Although QC-Trees can achieve a high compression ratio, it is still a fully materialized data cube. In this paper, we present an improved structure PMC, which allow us to partially materialize cells in a QC-Trees. There is a sharp contrast between our partially materialization algorithm and other extensively studied materialized view selection algorithms. If a view is selected in a traditional algorithm, then all cells in this selected view are to be materialized. Our algorithm, however, selects and materializes data by cells. Experiments results show that PMC can further reduce storage space occupied by the data cube, and can shorten the time for update the cube. Along with further reduced space and update cost, our algorithm can ensure a stable query performance.

## 1 Introduction

Using pre-aggregated data in data cube is an efficient method to improve query performances of data cubes. Usually, to attain better storage and update performance, only a part of views are materially stored in the database. Apparently, there exists performance difference between materialized view and non-materialized view. Moreover, to select materialized views, prior knowledge about the query model of users should be attained. This kind of knowledge, however, are usually not available.

QC-Trees is one of the most compress efficient structures of data cubes. This structure has significant merits. The more sparse the data cube is, the more storage space can be saved. Since typically high-dimension data cubes are sparse, a high compression ratio can be easily achieved.

In this paper, we present PMC and corresponding algorithms to select only a part of cells in QC-Tree to be materialized so that better space and update efficiencies are achieved. In contrast with traditional view materialization algorithms, cells (not views) are selected to be materialized. In this way, a low bound of query can be ensured by our structure. This low bound is regulated by a parameter of selection algorithm, which can be used for controlling the tradeoff between storage, update and query performances.

This paper includes following contributions:

Firstly, we present a structure PMC (Partially Materialized Cube), which can be seen as an extension or a generalization of QC-Tree. To our knowledge, our work is the first attempt to partially materialize data cube at the cell level.

Secondly, corresponding algorithms are presented, including selection of materialized cells, queries and update process for PMC. Besides the costs of storage and maintenance are reduced, the query performance can be stably ensured by a parameter of the selecting algorithm.

The paper is organized as follows: related work are reviewed in Section 2. In section 3, we present PMC structure and corresponding algorithms including select, update and query algorithms. Then in Section 4 we conduct experiments to evaluate PMC algorithm, while conclusions are presented in Section 5.

## 2 Related Work

Online analytical processing (OLAP) is an essential data analysis service and can provide critical insights into huge amount of application data. Data Cube, which was proposed by J. Gray[2], has been extensively applied to the implementation of OLAP. Data in a data cube can be stored in special data structures(MOLAP) or in tables of relational databases(ROLAP).

Using pre-aggregated data in data cube is an efficient method to improve query performance of OLAP. However, this method brings serious space and maintenance problems. Although this situation can be relived by materializing only a subset of all views, which has been extensively studied in recent years, this strategy results in significant difference of query performance between materialized views and non-materialized views. Another flaw of this strategy is that the cost model for selecting views often needs prior knowledge about how users query the data. However, it is very difficult to attain this knowledge.

For MOLAP systems, data cubes are typically stored in (sometimes compressed) data arrays[10]. Although this method can help to achieve better queries performance, the size of these data cube become more huge. A lot of papers tried to compress data cube by compressing the arrays[6], or only providing inaccurate answers for queries[1][7].

Recently several new structures, such as condensed cube[9], Dwarf[8], Quotient cube[4] and QC-Trees[5], are proposed to substantially reduce the size of the data cube. These structures have similar idea: when several different level aggregation values have the same certain characteristics, these structures can store them in one storage unit. In other words, they use one memory unit to present several cells in data cube, if these cell are computed by the same subset of the fact table's tuples.

In 2002, W. Wang[9] proposed the concept of BST(Base Single Tuple). If a cell  $SD$  in the data cube only covers one tuple  $r$  in the base table, then  $r$  is called the BST of  $SD$ . If  $r$  is a BST on some (preferably maximized)  $SD$ , the aggregation function should be applied only, and exactly once, to tuple  $r$ . Moreover, for every BST  $r$ , only one base tuple needs to be stored in the Condensed Cube.

Dwarf[8] has a similar and better idea. It adopts directed acyclic graph(DAG) to reduce prefix redundancy. To reduce suffix redundancy, it places Dwarf, several different level cells which can be aggregated from one set of base tables, in one storage unit.

In Quotient cube[4], the set of cells of a Data Cube are partitioned into certain number of equivalent classes. One equivalent classes, which contain one or some cell(s) all have the same certain characteristics, will be coalesced into one storage unit.

QC-Trees[5], which is the Quotient cube with *cover* equivalence, is one of the most efficient structures among them. For example, a cell  $c$  covers a base table tuple  $t$  whenever there exists a roll-up path from  $t$  to  $c$ . If the set of tuples in the base table covered by cell  $c$  and cell  $d$  are the same, then  $c$  and  $d$  are *cover* equivalent and their aggregation values are equal. In QC-Trees, both of them will be partitioned into one class. Another point of QC-Trees is that all cells in one class can be represented by the finest cell (called the *upper bound* of this class) in the class.

### 3 Partially Materialized Cells

In this section, we develop a data structure PMC(Partially Materialized Cells) to partially materialize QC-Tree at the cell level. Algorithms are also presented for the constructing , updating and querying of PMC. In this section, a part of algorithms are abridged for space reasons.

#### 3.1 Motivation

In the QC-Tree, a node represents one or several cell(s) whose aggregation is not NULL in the data cube. On the other hand, each non-NULL cell can find the only corresponding node in the QC-Tree.

We noticed that a node in QC-Tree may possess many children (by edges or links), whose labels may belong to one of several dimensions. Moreover, the value of the node<sup>1</sup> is equal to the aggregates of all children which belong to any one of dimensions. For example, in Fig. 1, node 1 has 6 children: node 2,8,11(by edge) and node 5, 7, 10 (by link). The labels of node 2,8 belong to dimension *Store*, while 5,11 belong to *Product* and 7,10 belong to *Season*. The value of node 1 and the aggregation of all its children on any one dimension are the same: 27.

The point is, since the value of the node is equal to the aggregation of its children, the value of the node needn't to be stored if the cost of aggregating from its children is trivial. When querying, we can retrieve the value by temporarily aggregating the values of its children.

Thus, much of storage space can be saved. Moreover, the maintenance performance of the data cube can be significantly improved.

---

<sup>1</sup> In the following part of this paper, when we mention “the value of a node” , we are referring to “the aggregation value of the cell which is correspond to the node ”.



Store	Product	Season	Price
S1	P1	s	\$6
S1	P2	s	\$12
S2	P1	f	\$9

Table 1. The base table

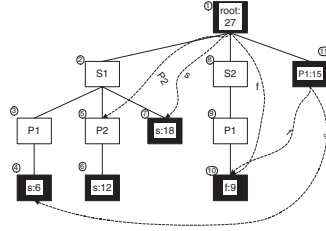


Fig. 1. The QC-Tree for Table 1

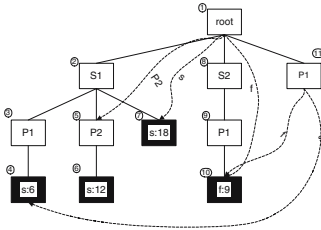


Fig. 2. The PMC for Table 1

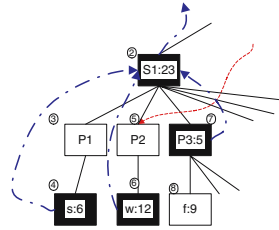


Fig. 3. A part of a slight complex PMC

For example, in a marketing management data warehouse, data are collected under the schema `sales(Store, Product, Season, Price)`. The base table, which holds the sales records, is shown in Fig. 1. Attributes `Store`, `Product` and `Season` are *dimensions*, while attribute `Price` is a *measure* and  $SUM(Sale)$  as the aggregate function.

The *data cube*, with `Store`, `Product` and `Season` as dimensions and  $SUM(Sale)$  as an aggregate function, is a set of results returned from being grouped by each subset of these three dimensions. Each group-by corresponds to a set of cells, described as tuples over the group-by dimensions.

A QC-Tree example for Table 1 is shown in Fig. 1, while Fig. 2 shows the PMC from the same base data, where we set the threshold for query costs of PMC to 2. Non-materialized nodes are marked with white boxes, while materialized nodes are marked with black boxes. The edges are denoted by solid lines, the link by the read thin broken lines. Compared to Fig. 1, node 1 and node 11 in Fig. 2 are no longer to be materialized because the cost of computing each of their values is not exceed 2. For example, A query for node 11 can be answered by aggregating values of two other nodes: node 4 and node 10. In fact, QC-Tree can be seen as the special case of PMC with threshold= 1.

Upon update problem, we have following considerations:

For a single cell, if it is to be updated, logically all cells roll up from it should be updated to ensure consistency. QC-Trees can achieve a better update performance than normal data cube just because a cell in QC-Trees may present several coarser cells whose cover set are identical with it. When the base cell is updated, values of those coarser cells needn't be modified because they are linked to the base cell. In other words, one modified action on the base cell also

update all other cells with the same cover set, which is the reason why update operation can be reduced in QC-Trees.

So the more other coarser cells are linked to the cell, the more the update cost is reduced. In QC-Trees, an aggregated cell is only linked to one cell when both cells' cover sets are identical. In PMC, a non-materialized node can be linked to one or several materialized node(s). As a result, PMC goes a further step and makes more cells non-materialized.

Therefore, there are less nodes are to be update in PMC than in QC-Trees when maintaining. As a result, PMC can achieve better update performance than QC-Tree.

For example, in Fig. 1, if the value of node 10 are changed, the values of node 1 and node 11 should also be updated . However, in Fig. 2, such a change on node 9 will not cause a update on node 1 and node 11.

### 3.2 The Structure of PMC

PMC is similar to QC-Tree , except that

- each node in PMC has an additional value *cost*, which denotes the cost of retrieving the value of the node. For materialized node, the cost is set to 1. For non-materialized node, the cost is set to the time of access its descendants' values when aggregating through edges and(or) links.
- all non-materialized nodes don't store their value.
- (optional) each materialized node has a number of update-links which point to its direct materialized descendants. These update-link are used for accelerating the process of update.

In this paper, the value of *cost* of a node is defined as the number of accessing (materialized) node(s)' value(s) in the course of get the value of the node. For a materialized node, since its value is stored in the node itself, its cost is 1. To get the value of a non-materialized node, we need drill down and get the value by aggregating its descendants' values. Therefor, if a node has a large *cost*, we will spend much on retrieving its value when answering queries. In order to avoid this situation, we have this kind of nodes materialized so that we can retrieve any node's value easily.

Fig. 3 shows a part of a more complex PMC, where the update-links are marked by the blue thick broken lines.

Without loss of generality, we assume the type of aggregate function is SUM. In fact, other common aggregate functions like COUNT, MAX, MIN can also be the aggregate function of PMC, so does AVG, which can be denoted as SUM/COUNT.

### 3.3 Construction of PMC

The construction of PMC is in four steps.

- Firstly, PMC is constructed like a QC-tree. The difference is that our algorithm needn't compute the aggregate value for each temporary class. In a

```

Function PointQuery(CurrentNode)
local variable: localvalue
initiate localvalue;
if CurrentNode is materialized then
    localvalue=CurrentNode.Vaule;
else
    Find the drill dimension D whose query cost is minimum;
    for each child CD along dimension D of CurrentNode do
        localvalue=aggregate(localvalue , PointQuery(CD));
    endfor;
endif;
return localvalue;

Function Select(CurrentNode)
for every edge's child nodes cd of CurrentNode do
    Select(cd);
endfor;
//re-computer CurrentNode's cost
if cost(CurrentNode) > Threshold then
    SetMaterialized(CurrentNode);
    BuildUpdateLinks(CurrentNode,CurrentNode);
    compute and store CurrentNode's aggregation value;
    /*now the cost of CurrentNode becomes 1*/
endif;
return(cost(CurrentNode));

```

Fig. 4. Function *PointQuery* and Function *Select*

QC-Tree, the aggregates of all temporary classes have to be computed, and a majority of them are repetitional and useless in the course of constructing QC-Tree.

- *Cost* of each node is initiated in the second step. In this step, only leaf nodes of the tree are materialized, so the *cost* of a node is in fact equal to the number of the base tuple covered by the node.
- In third step, function *select* is executed for the tree and selects the set of nodes to be materialized. The parameter  $C_{threshold}$  of *select* is used to control this process. Those nodes whose *cost* are larger than  $C_{threshold}$  will be materialized so that their *costs* are reduced to 1. By doing so, the cost of query of all cells in PMC can be limited below  $C_{threshold}$ .
- Finally, we perform *BuildUpdateLinks* to build update links for materialized nodes. The update links are set among materialized nodes. Each materialized node has update links which point to those nodes whose value is computed from this node. Of course, if we want build a PMC without update-links, we can jump over this step.

In Fig. 4, we present *select* function, which selects a set of nodes and then materializes them so that there is no node whose *cost* exceeds  $Cost_{threshold}$  in PMC. It's interesting that if this  $Cost_{threshold}$  is set to 1, the corresponding PMC is identical with the QC-Trees. It shows that QC-Tree is in fact the special case of PMC with  $Cost_{threshold} = 1$ .

For the sake of brevity, we do not show the detail of *BuildUpdateLinks*.

### 3.4 Query

**Point Query** The answering for point query is rather easy. If the node is materialized, we'll get the answer from the node immediately. Otherwise, the answer will be retrieved by computing its descendants' value. It's clear that the cost of this computation will not exceed  $Cost_{threshold}$ . Algorithm for point query is shown in Fig. 4.

### 3.5 Maintenance of PMC

To the PMC without update-links, the process of maintenance is similar to that of QC-Tree. The difference is that only materialized nodes need to be *updated* in PMC. As a result, less temporary "*update*" classes are produced in PMC and thus the maintenance of PMC should be slight faster than that of QC-Tree.

If we want a better update performance, update-links can be build for acceleration. In following part of this subsection, we'll address the issues concerned with the maintenance of the PMC with update-links.

There are three types of maintaining operations on the base table: insert, delete and update. Therefore, the change of source data can be represented simply by tuples inserted into, deleted from, and updated in the base table.

**Update.** For a newly coming tuple, if there exists a corresponding node in PMC, then a update operation starts. It's clear that the corresponding node is a leaf node. From subsection 3.3, we know that leaf nodes are materialized. Thus, The node is materialized and has update links.

In PMC, values of nodes are update along update links. When the value is to be updated, a (materialized) node will call the update functions of those nodes which are pointed by its update links. The update value for the current node will be passed as a parameter along these update functions. Thus, computing aggregate values of newly coming tuples for temporary classes is avoided.

If execute the update function for a leaf node, all its materialized ancestors will be updated.

If PMC is updated tuple by tuple, we can simply call the update functions of corresponding leaf nodes to accomplish the update of PMC.

**Batch Update.** In theory, batch update can be accomplished by perform *update* operations tuple by tuple. This strategy, however, is inefficient. In this way, if a node is updated for  $t$  times, then all its ancestors' value are bound to be modified for  $t$  times. In other words, if a node covers  $n$  newly coming tuples, the update function of this node will be executed for  $n$  times in the course of maintenance.

Therefore, for batch update, we adopt another strategy. Fig. 5 shows the function *BatchUpdate*. Each materialized node has a variable *tempV*, and it is initiated as 0. When the update function of this node is executed, the parameter *Value*, which is passed from one of its descendant, will used to cumulatively modify (aggregate) *tempV* instead of the value of this node. Moreover, other update functions will not be called by this update function. Only when it is the last time the update function of this node is executed, will *tempV* be used

```

Function BatchUpdate(CurrentNode, Value)
    CurrentNode.Countto_update--;
    if CurrentNode.Countto_update > 0 then
        tempV+=Value;
    else
        SetNewValue(CurrentNode, CurrentNode.tempV);
        for every UpdateLink from CurrentNode do
            Update(UpdateLink.TargetNode, CurrentNode.tempV);
        endfor;
    end if;

```

**Fig. 5.** Function *BatchUpdate*

to change the value of this node. And then, its ancestors' update functions are called along update links and take *tempV* as parameter.

This strategy has following merits:

1. No matter how many times will a node be updated , all its ancestors' update functions will be executed only *once*.
2. Since values for update in PMC are passed along update links, we needn't compute aggregate values of temporary classes in  $\Delta$ DFS. Thus, a lot of data access and computations are avoided. Compared to it, computation for the initial value of *Count<sub>to\_update</sub>* needn't access the value of raw data and its complexity can be neglected.

**Insertion and Deletion.** When there is no tuple in the base table such that it has the same dimension values as the newly coming tuple, new classes should be split from a old class, or be created. In this paper, we view an insertion operation as the combination of (possibly) a node creation and a update operation.

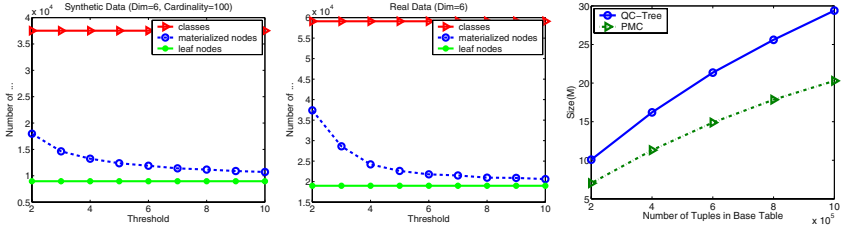
Similar to insertion operation, we view a deletion operation as the combination of an update operation and possible mergence or deletion of old node(s).

Due to the limitation of space, we skip the detail of this part.

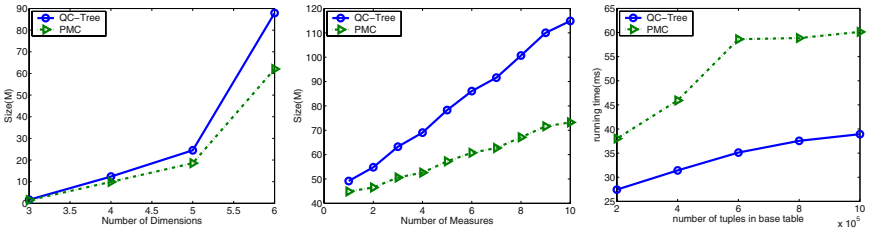
## 4 Experimental Results

In this section, synthetic and real data sets are used for examining the algorithms we presented in this paper. We focus on three main performance issues(storage, queries answering and maintenance) of PMC and compare them with those of the QC-Trees.

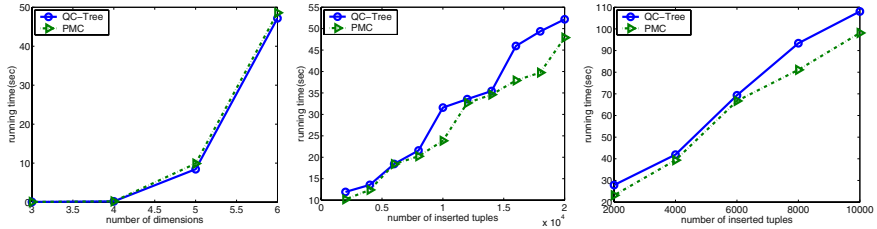
There are two classes of data sets used in the experiments. The first one is synthetic data with Zipf distribution. The synthetic data set contains 1,000,000 tuples and 6 dimensions, the cardinality of each dimension is 100. The other one is the real data set containing weather conditions at various stations on land for September 1985[3]. The weather data set contains 1,015,367 tuples and 6 dimensions. The attributes are as follows with cardinalities listed in parentheses: longitude (352), solar-altitude (179), latitude(152), present-weather (101),



(a) threshold vs. storage (synthetic data) (b) threshold vs. storage (real data) (c) number of base tuples vs. storage (synthetic data)



(d) number of dimensions vs. storage (real data) (e) number of measures vs. storage (real data) (f) Query Performance (synthetic data)



(g) Query Performance (real data) (h) Maintenance Performance (synthetic data) (i) Maintenance Performance (real data)

**Fig. 6.** Experimental Results

weather-change-code(10), and hour (8). All experiments are running on a Pentium 4 PC with 256MB main memory and the operation system is Windows XP.

### 4.1 Storage

In QC-Tree, the number of data units is equal to the number of classes, while in PMC it is equal to the number of materialized node. Fig. 6(a) and 6(b) illustrate the value of thresholds versus the number of data storage units needed. The results show that small thresholds (like 2 or 3) can achieve satisfying balance

between storage and query. In Fig. 6(a) and 6(b), number of tuples in the base table is 20000 .

Fig. 6(c) shows the results of storage vs. number of base tuples, while Fig. 6(d) shows storage vs. the number of dimensions and Fig. 6(e) shows storage vs. the number of measures. The results illustrate that PMC can compresses the data cube more efficiently than QC-Tree.

## 4.2 Queries Answering

In this experiment, we compared query answering performance of PMC and QC-tree. We randomly generated 1,000 point queries and range queries on each data set. Fig. 6(f) and 6(g) show the times for answering point queries on synthetic data and range queries on real data separately. Other experiments have similar results.

The results show that the query performance of PMC is only a little worse than that of QC-Tree.

## 4.3 Maintenance

To test the performance of the incremental maintenance of PMC, we generated different size of data, inserted them into a size-fixed PMC. The tuples are batch inserted. The results are shown in Fig. 6(h) and 6(i).

The results show that the maintenance performance of PMC is better than QC-Tree. The reasons are: (1). our algorithm will conduct less aggregate operation when generating temporary classes; (2)there are less materialized nodes to be update in PMC.

## 5 Conclusions

In this paper, we present algorithms to partially materialize cells in the data cube. Compared with extensively studied view selection algorithms, our algorithms deal with cells in compressed structure of data cube. Along with reduced space and update cost, our algorithm can ensure a stable query performance.

Our algorithms can achieve a better tradeoff between storage, update and query performances of a Data cube. As a example in this paper, our algorithms are applied to QC-tree. In fact, other cell-level structure, such Dwarf or Condensed Cube, can also adapt our algorithms with appropriate modifications.

## References

1. K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In Proc, SIGMOD, pages 359-370, 1999.
2. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In Proc, ICDE, pages 152-159, 1996.

3. C. Hahn et al. Edited synoptic cloud reports from ships and land stations over the globe, 1982-1991. [cdiac.est.ornl.gov/ftp/ndp026b/SEP85L.Z](http://cdiac.est.ornl.gov/ftp/ndp026b/SEP85L.Z), 1994.
4. Laks V. S. Lakshmanan, Jian Pei, Jiawei Han, Quotient Cube: How to Summarize the Semantics of a Data Cube. In Proc, VLDB, pages 778-789, 2002.
5. Laks V. S. Lakshmanan, Jian Pei, Yan Zhao: QC-Trees: An Efficient Summary Structure for Semantic OLAP. In Proc, SIGMOD, pages 64-75, 2003.
6. J. Li, Y. Li, J. Srivastava: Efficient Aggregation Algorithms on Very Large Compressed Data Warehouses. *J. Comput. Sci. Technol.* 15(3): 213-229 (2000).
7. J. Shanmugasundaram, U. Fayyad, and P. Bradley. Compressed data cubes for olap aggregate query approximation on continuous dimensions. In Proc, SIGKDD, pages 223-232, 1999.
8. Y. Sismanis, A. Deligiannakis, N. Roussopoulos, Y. Kotidis, Dwarf: shrinking the PetaCube. In Proc, SIGMOD, pages 464-475, 2002.
9. W. Wang, H. Lu, J. Feng, and J. Yu. Condensed Cube: An Effective Approach to Reducing Data Cube Size. In Proc, ICDE, pages 155-165, 2002.
10. Y. Zhao, P. Deshpande, and J. Naughton, An Array-Based Algorithm for Simultaneous Multidimensional Aggregates. In Proc, SIGMOD, pages 159-170, 1997.



# Progressive Ranking of Range Aggregates\*

Hua-Gang Li, Hailing Yu, Divyakant Agrawal, and Amr El Abbadi

University of California, Santa Barbara, CA 93106, USA  
{huagang, hailing, agrawal, amr}@cs.ucsb.edu

**Abstract.** Ranking-aware queries have been gaining much attention recently in many applications such as search engines and data streams. They are, however, not only restricted to such applications but are also very useful in OLAP applications. In this paper, we introduce *aggregation ranking* queries in OLAP data cubes motivated by an online advertisement tracking data warehouse application. These queries aggregate information over a specified range and then return the ranked order of the aggregated values. They differ from range aggregate queries in that range aggregate queries are mainly concerned with an aggregate operator such as SUM and MIN/MAX over the selected ranges of all dimensions in the data cubes. Existing techniques for range aggregate queries are not able to process aggregation ranking queries efficiently. Hence, in this paper we propose new algorithms to handle this problem. The essence of the proposed algorithms is based on both ranking and cumulative information to progressively rank aggregation results. Furthermore we empirically evaluate our techniques and the experimental results show that the query cost is improved significantly.

## 1 Introduction

Traditionally, databases handle unordered sets of information and queries return unordered sets of values or tuples. However, recently, the ranking or ordering of members of the answer set has been gaining in importance. The most prevalent applications include search engines where the qualifying candidates to a given query are ordered based on some priority criterion [3]; ranking-aware query processing in relational databases [14,11,2,10,4,7]; and network monitoring where top ranking sources of data packets need to be identified to detect denial-of-service attacks [1,9]. Ranking of query answers is not only relevant to such applications, but is also crucial for OnLine Analytical Processing (OLAP) applications. More precisely ranking of aggregation results plays a critical role in decision making. Thus, in this paper, we propose and solve aggregation ranking over massive historical datasets.

As a motivating example, consider an online advertisement tracking company<sup>1</sup>, where each advertiser places its advertisements on different publishers' pages, e.g., CNN and BBC. In general an advertiser is interested in identifying the "top" publishers in terms of total sales or number of clicks during a specific time period. For instance,

---

\* This research is supported by the NSF grants under IIS-23022, CNF-0423336, and EIA-00-80134.

<sup>1</sup> The proposed research is motivated by a real need for such type of algorithmic support in an application that arises in a large commercial entity, an online advertisement tracking company.

during a period of last 30 days while a particular advertisement campaign was conducted, or during the period of 15 days preceding the new year. Such an advertising company would need to maintain a data warehouse which stores data cube information regarding the sales (or clicks) of the various publishers and advertisers, and where an advertiser would like to ask queries of the form: “*find the top-10 publishers in terms of total sales from Dec 15, 2003 to Dec 31, 2003*”. Based on existing techniques, first the total sales from Dec 15, 2003 to Dec 31, 2003 for each publisher needs to be computed. Then the total sales for all publishers are sorted to identify the top-10 publishers. We refer to such queries as *aggregation ranking*, since they aggregate information over a specified range and then return the ranked order of the results. An alternative example is in the context of the stock market data. For example, given the trade volume of each stock, an analyst might be interested in the top trades during a certain period.

The problem of aggregation ranking is similar and yet differs from many related problems which have been addressed by the database and related research communities. We concentrate on the online analysis of massive amounts of data, which is similar to range aggregate queries prevalent in data warehouses. However, we are concerned with ranking of aggregated values over dimensions while prior research work on range aggregate queries has mainly concentrated on a single aggregate operator such as SUM and MIN/MAX over selected ranges of dimensions [6,13]. To the best of our knowledge, this paper is the first attempt to address the ranking of aggregation in the context of OLAP applications. Our approach differs from the data stream research related to the TOP- $k$  operations [1,9,5] since the data is not continuously evolving. Moreover, queries in data streams are interested in more recent data. In contrast, our aggregation ranking queries can involve data in any arbitrary time range. In the context of relational databases, Bruno et al. [2] proposed to evaluate a top- $k$  selection by exploiting statistics stored in a RDBMS. Ilyas et al. [11,10] proposed a new database operator, top- $k$  join, and efficiently implemented it using available sorting information of joined relations. This work addresses the optimization of top- $k$  selection and join operations in the context of relational databases. Our work, however, targets aggregation ranking queries in OLAP applications. In multimedia systems, Fagin [8] introduced ranking queries that combine information from multiple subsystems. Fagin’s algorithm can be directly applied if aggregates at multiple granularities (e.g. day, month, year) are considered. In particular aggregates on any specified range can be obtained by additions of multiple involved lists at different granularities. However when the number of involved lists grows large, Fagin’s algorithm tends to have a linear cost while our proposed algorithms in this paper always involve only two lists with sublinear cost. Furthermore, the framework in [8] is indeed useful for reasoning the correctness of our algorithms for aggregation ranking queries and therefore we adapt it to our context.

The rest of the paper is organized as follows. Section 2 gives the model and a motivating example. In Section 3, we present a new cube representation. Then we incrementally develop three different techniques for answering aggregation ranking queries in the following three sections, each of these improves a previous one. In Section 7 we empirically evaluate our proposed techniques and present the experimental results. Conclusions and future research work are given in Section 8.

## 2 Model and Motivating Example

In this paper we adopt the data cube [12] as a data model, A data cube can be conceptually viewed as a hyper-rectangle which contains  $d$  dimensions or *functional attributes* and one or more *measure attributes*. Dimensions describe the data space, and measure attributes are the metrics of interest (e.g., sales volume). In particular, each cell in a data cube is described by a unique combination of dimension values and contains the corresponding value of the measure attribute. To introduce aggregation ranking queries, we assume that among the  $d$  functional attributes of a data cube, one of the functional attributes,  $A_r$ , is the *ranking functional attribute* and the rest  $d - 1$  functional attributes are the *range functional attributes*. An aggregation ranking query specifies ranges over the  $d - 1$  range functional attributes and requests a ranking of the values of the ranking functional attribute  $A_r$  based on the aggregated values of the measure attribute after applying some type of aggregation over the specified ranges.

For instance, using the online advertisement tracking company example, we consider a 2-dimensional data cube SALES for a particular advertiser  $a$ , which has Publisher as the ranking functional attribute, Date as the range functional attribute and Sales as the measure attribute. Each cell in this data cube contains the daily sales of advertiser  $a$  through the advertisements placed on a publisher  $p$ 's website. Fig. 1(a) shows an example SALES data cube. A particular type of aggregation ranking query of interest to advertiser  $a$  in the SALES data cube is “find the top- $k$  publishers in terms of total sales from day  $D_s$  to  $D_e$ ”, and is specified as  $AR(k, D_s, D_e)$  for simplicity. The shaded area in Fig. 1(a) shows an instance of such a query from day  $D_3$  to  $D_6$ . Answering this kind of aggregation ranking queries with SUM operator efficiently is the focus of this paper. A basic way to answer such a query is to access each selected cell in the data cube to compute the total sales for each publisher within the time range from  $D_s$  to  $D_e$ . Then we sort the aggregated values to obtain the top- $k$  publishers. Since the number of involved cells is usually large, and data cubes are generally stored on disks,

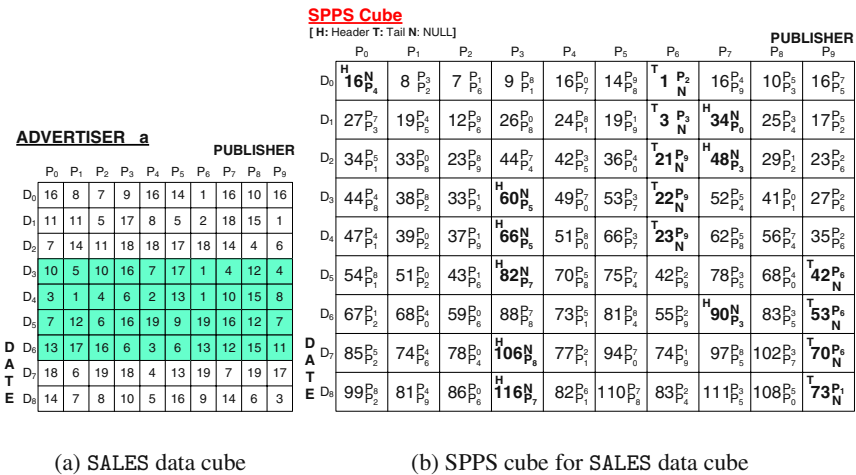


Fig. 1. A SALES data cube and its SPPS cube

this will result in significant overhead. Also online sorting entails significant time overhead if there is a large number of publishers per advertiser. This in turn will impact the response time of interactive queries negatively.

### 3 Sorted Partial Prefix Sum Cube

In order to process aggregation ranking queries efficiently, we propose to use cumulative information maintained for each value of the ranking attribute  $A_r$  along the time dimension. This is based on the prefix sum approach [6] which can answer any range aggregate query in constant time. Furthermore we pre-sort the values of  $A_r$  for each time unit based on the cumulative information. Hence a new cube presentation, *Sorted Partial Prefix Sum Cube* (SPPS cube in short), is developed. SPPS cube has exactly the same size as the original data cube. For simplicity of presentation, we will use the online advertisement tracking company example to explain our data structures and algorithms. The proposed algorithms can be generalized to handle data cubes with any arbitrary number of dimensions in a straightforward manner. An SPPS cube for the SALES data cube contains cumulative information along the DATE dimension for each publisher and daily order information along the PUBLISHER dimension. Each cell in the SPPS cube, indexed by  $(P_i, D_i)$ , maintains the following three types of information:

- PPSUM (Partial Prefix Sum): total sales for publisher  $P_i$  within the time range from  $D_0$  to  $D_i$ , i.e., cumulative sum Since the initial time of the SALES data cube.
- PPC (Pointer to Previous Cell): a pointer to a cell in the same row of the SPPS cube which contains the least value no less than  $\text{SPPS}[P_i, D_i].\text{PPSUM}$ ; if such a pointer does not exist, PPC is set to NULL.
- PNC (Pointer to Next Cell): a pointer to a cell in the same row of the SPPS cube which contains the largest value no greater than  $\text{SPPS}[P_i, D_i].\text{PPSUM}$ ; if such a pointer does not exist, PNC is set to NULL.

PPC and PNC for all cells in a given row or time unit,  $D_i$ , maintain a *doubly linked list* in decreasing order of PPSUM. We refer to this list as  $\text{PPSUM}(D_i)$ . In addition we maintain two pointers pointing to the *header* and the *tail* of each doubly linked list for the SPPS cube. The header is the top ranked publisher based on the cumulative sales from the initial date of the SALES data cube and the tail is the bottom ranked publisher.

Fig. 1(b) shows the SPPS cube for the SALES data cube in Fig. 1(a). Each cell contains PPSUM, PPC, PNC information in the form of  $\text{PPSUM}_{\text{PNC}}^{\text{PPC}}$ . Also for presentation simplicity, we use the publisher index for pointers PPC/PNC. Note that the preprocessing of SPPS cube is offline, which is typical in real data warehousing applications. Space and offline processing are usually sacrificed for online interactive query processing.

### 4 Complete Scan Algorithm

We first present a simple algorithm, *complete scan*, to process aggregation ranking queries by using the pre-computed cumulative information in SPPS cubes. Given a query  $AR(k, D_s, D_e)$ , we need to obtain the total sales  $\text{SUM}(P_i, D_s, D_e)$  for each publisher  $P_i$  from  $D_s$  to  $D_e$ . This can be computed from  $\text{PPSUM}(D_i)$  maintained for each publisher  $P_i$  in the SPPS cube, which is actually given by the following subtraction:

$$\text{SUM}(P_i, D_s, D_e) = \text{SPPS}[P_i, D_e].\text{PPSUM} - \text{SPPS}[P_i, D_s - 1].\text{PPSUM}.$$

Collecting the total sales of all publishers between  $D_s$  and  $D_e$  together, we get a list denoted by  $\text{SUM}(D_s, D_e) = \{\text{SUM}(P_0, D_s, D_e), \dots, \text{SUM}(P_{n-1}, D_s, D_e)\}$ . Then the top- $k$  publishers during the period  $(D_s, D_e)$  can be easily extracted from the list  $\text{SUM}(D_s, D_e)$  as follows. Take the first  $k$  publishers from the SUM list, sort and store them into a list called *list-k*. For each publisher in the sum list ranging from  $k + 1$  to  $n$ , insert it into *list-k*, then remove the smallest publisher from *list-k*. Therefore the publishers in the final *list-k* are the top- $k$  publishers. The query cost is  $O(n + n \log k)$ . If  $k$  is a constant or  $k$  is much smaller than  $n$  ( $k \ll n$ ), the query cost is linear.

Note that the cost of the query is independent of the query range in the time dimension and is linearly dependent on the total number of publishers. Since the data cube is stored on disks, the cost of retrieving every publisher's information from disk can be relatively high. Furthermore an online advertisement tracking data warehouse serves a large number of advertisers at the same time. Thus the delay may not be acceptable for analysts who prefer interactive response time. In the next two sections, we extend the complete scan algorithm to improve the query cost by exploiting the ranking information maintained in the SPPS cube to minimize the number of publishers scanned.

## 5 Bi-directional Traversal Algorithm

In the complete scan algorithm, the first step computes the total sales for each publisher in a given time range, for which the best time complexity is linear. In order to reduce the total query cost, we need to avoid computing the entire SUM list. This is the premise of the bi-direction traversal algorithm discussed in this section.

The problem of evaluating aggregation ranking queries now reduces to the problem of combining two lists of ordered partial prefix sums corresponding to the given time range  $(D_s, D_e)$ , i.e.,  $\text{PPSUM}(D_s - 1)$  and  $\text{PPSUM}(D_e)$  respectively. Intuitively, for a given query  $AR(k, D_s, D_e)$ , the publishers which are in the query result must have relatively larger values in list  $\text{PPSUM}(D_e)$  and relatively smaller values in list  $\text{PPSUM}(D_s - 1)$ . Thus, instead of computing the entire list of  $\text{SUM}(D_s, D_e)$ , we may only need to compute the total sales of publishers which have higher ranking in  $\text{PPSUM}(D_e)$ , and lower ranking in  $\text{PPSUM}(D_s - 1)$  as long as the number of these publishers is large enough to answer the aggregation ranking query. Based on this intuition, we design the *bi-directional traversal algorithm* shown in Algorithm 1.

In the bi-directional traversal algorithm, we extract publishers concurrently from list  $\text{PPSUM}(D_e)$  in decreasing order (starting from the header of  $\text{PPSUM}(D_e)$  down to the tail) into a list denoted by  $L_e$ , and from list  $\text{PPSUM}(D_s - 1)$  in increasing order (starting from the tail of  $\text{PPSUM}(D_s - 1)$  up to the header) into another list denoted by  $L_s$ , until the number of publishers in the intersection of their output sets  $L_s \cap L_e$  is no smaller than  $k$ . Hence scanning all the publishers is avoided. Then calculate the total sales of all publishers in  $L = L_s \cup L_e$ . Finally, compute the top- $k$  publishers in  $L = L_s \cup L_e$  based on their total sales during  $(D_s, D_e)$ . These top- $k$  publishers are actually the answer to the given query. The bi-directional traversal algorithm improves the processing cost of  $AR(k, D_s, D_e)$  to  $O(\sqrt{n})$  with arbitrarily high probability if the two lists  $\text{PPSUM}(D_s - 1)$  and  $\text{PPSUM}(D_e - 1)$  are independent and  $k$  is much smaller than

**Algorithm 1** Bi-directional Traversal Algorithm

---

```

1: Input:
2:  $AR(k, D_s, D_e)$ ;
3: Procedure
4:  $L_s = \phi, L_e = \phi$ ;
5:  $POINTER_s = \text{Tail of PPSUM}(D_s - 1)$ 
6:  $POINTER_e = \text{Header of PPSUM}(D_e)$ ;
7: while  $|L_s \cap L_e| < k$  do
8:    $L_s = L_s \cup POINTER_s.\text{publisher}$ ;
9:    $POINTER_s = POINTER_s.PPC$ 
10:   $L_e = L_e \cup POINTER_e.\text{publisher}$ ;
11:   $POINTER_e = POINTER_e.PNC$ 
12: end while
13: for each publisher  $P$  in  $L = L_s \cup L_e$  do
14:   Compute the total sales in  $[D_s, D_e]$  by  $SPPS[P, D_e].PPSUM - SPPS[P, D_s - 1].PPSUM$ ;
15:   Insert  $P$  into set  $R$ ;
16:   if  $|R| > k$  then
17:     Remove  $P_i$  from  $R$  if its  $SUM(P_i, D_s, D_e)$  is smaller than all other publishers in  $R$ ;
18:   end if
19: end for
20: End Procedure
21: Output:  $R$ ;

```

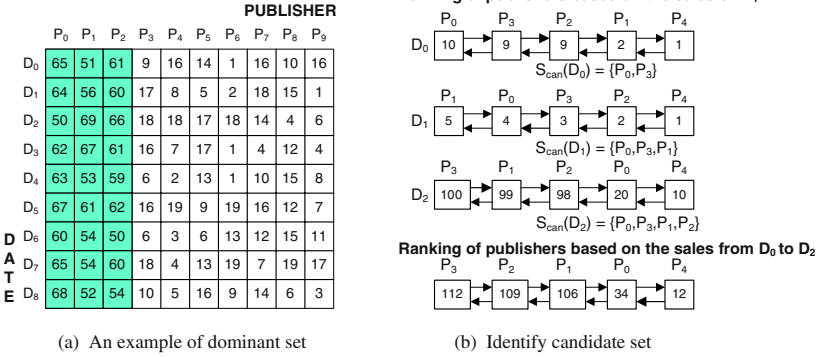
---

$n$ . Due to the space limit, please refer to [15] for further details of query cost analysis and the correctness proof of the bi-directional algorithm.

## 6 Dominant-Set Oriented Algorithm

The bi-directional traversal algorithm can answer a query  $AR(k, D_s, D_e)$  in  $O(\sqrt{n})$  with arbitrarily high probability for  $n$  publishers if the two lists  $PPSUM(D_s - 1)$  and  $PPSUM(D_e)$  are independent. Unfortunately in most real applications, this is not the case. For example, considering the online advertisement tracking data warehouse application, the two lists are independent if the probability of daily sales is not dependent on a specific publisher, i.e., if all publishers have similar and independent degree of popularity. However, in real world, some publishers are usually more popular than others. Thus the daily sales obtained through the advertisements placed on those publishers are much more than that of other publishers. Under such circumstances, the cumulative sales in lists  $PPSUM(D_s - 1)$  and  $PPSUM(D_e)$  for a publisher may not be completely independent. Therefore the probability that the query cost is  $O(\sqrt{n})$  becomes low. In particular, the worst case could happen when the two lists have almost the same set of publishers that always have the most daily sales. Fig. 2(a) shows such an example, where publishers  $P_0, P_1$ , and  $P_2$  always have more daily sales than the rest of the publishers. Given any query  $AR(k, D_s, D_e)$  over the data cube shown in Fig. 2(a), by using the bi-directional traversal algorithm, in order to get  $L_s \cap L_e \geq k$ , the number of publishers in  $L_s \cup L_e$  can be up to  $n$ . As a result, the query cost is almost linear. This is mainly because the bi-directional traversal algorithm is unable to minimize the size of a superset of the top- $k$  publishers efficiently in the presence of correlation among publishers and skewed distributions.

Hence, our goal now is to optimize the bi-directional traversal algorithm by pruning the search space in list  $PPSUM(D_s - 1)$ . In order to do that, we need to identify the candidates for an aggregation ranking query. Without any doubt, dominant publishers



**Fig. 2.** An example of dominant set and how to identify candidate set

usually dominate the top- $k$  slots and need to be considered in the candidate set. However some variations may occur, i.e., some non-dominant publishers may become dominant. Hence we need to identify such a set of candidates that may include the answer to an aggregation ranking query, for which we assume that all aggregation ranking queries  $AR(k, D_s, D_e)$  request a value of  $k$  no larger than  $k_{max}$  which is the maximum value of  $k$  specified in any aggregation ranking query. This is a realistic assumption, since advertisers are usually interested in a small number of publishers, especially those with a relatively high performance. We, therefore, assume that  $k_{max} \ll n$  and  $k_{max}$  is an application-dependent and user-defined parameter. We now introduce the notation of the *candidate set* for a day  $D_i$ , denoted as  $S_{can}(D_i)$ .  $S_{can}(D_0)$  is initialized to contain the top- $k_{max}$  publishers on the first day of the SALES data cube.  $S_{can}(D_i)$  contains all publishers in  $S_{can}(D_{i-1})$  and all publishers which are ranked on day  $D_i$  above any publisher in  $S_{can}(D_0)$  as well. We observe that  $S_{can}(D_{i-1}) \subseteq S_{can}(D_i)$ .

Consider the following example: assume there are 5 publishers  $P_0, P_1, P_2, P_3$  and  $P_4$  as shown in Fig. 2(b). We have the sales tracking information for three days  $D_0, D_1$  and  $D_2$ . Let  $k_{max}$  be 2.  $P_0$  and  $P_3$  ranked top-2 on day  $D_0$ . Hence  $S_{can}(D_0) = \{P_0, P_3\}$ . On day  $D_1$ ,  $P_1$  is ranked above  $P_3$  and  $P_3 \in S_{can}(D_0)$ , therefore,  $S_{can}(D_1) = \{P_0, P_3, P_1\}$ . Similarly  $S_{can}(D_2) = \{P_0, P_3, P_1, P_2\}$ . It is possible that a publisher which is ranked above any publisher in  $S_{can}(D_0)$  on day  $D_i$  could have a large total sales within some time range  $(D_s, D_i)$ . For example,  $P_2$  ranked top-2 in terms of total sales within  $(D_0, D_2)$ . Moreover, since  $P_4$  does not have a higher rank than the publishers in  $S_{can}(D_0)$  for any day, it is impossible to be a top-2 publisher for any aggregation ranking query.  $S_{can}(D_i)$  is a superset of the top- $k$  publishers for a given aggregation ranking query  $AR(k, D_s, D_i)$ , and the correctness is given in the following assertion<sup>2</sup>.

**Assertion 1.** For a given aggregation ranking query  $AR(k, D_s, D_e)$ , all the qualifying publishers must be contained in the candidate set for day  $D_e$ ,  $S_{can}(D_e)$ .

From Assertion 1, we know that in order to answer a given query  $AR(k, D_s, D_e)$ , we need to consider all the publishers in  $S_{can}(D_e)$ . A straightforward solution is to

<sup>2</sup> Please refer to [15] for proof.

obtain for each publisher  $p \in S_{can}(D_e)$  its prefix sum of sales from list PPSUM( $D_s - 1$ ) and list PPSUM( $D_e$ ). However this requires random accesses to both lists which results in a lot of random I/Os. In order to reduce the random accesses as well as consider all publishers in  $S_{can}(D_e)$ , we need to track the maximum index of all publishers in  $S_{can}(D_e)$  in list PPSUM( $D_s - 1$ ). We refer to this maximum index as the *pruning marker*. Note that the indices of cells in a list are in increasing order from the header to the tail. The header has an index of 0 and the tail has an index of  $n - 1$ . All publishers after the pruning marker in list PPSUM( $D_s - 1$ ) will be pruned as they do not qualify to be top- $k$  publisher candidates, and hence the search space is reduced.

However it is not efficient to compute the pruning marker online since finding the index of each publisher  $P \in S_{can}(D_e)$  in list PPSUM( $D_s - 1$ ) requires access to its corresponding cell in the SPPS cube. This can again degrade performance, especially when the size of  $S_{can}(D_e)$  is large. Since  $S_{can}(D_s - 1)$  is a subset of  $S_{can}(D_e)$ , we can pre-process the publishers in  $S_{can}(D_i - 1)$  for each date  $D_i$  and store the index corresponding to the smallest ranked publishers in  $S_{can}(D_i - 1)$ , and then process the remaining publishers for a given query. Hence for each day  $D_i$ , in addition to  $S_{can}(D_i)$ , we maintain the maximum index in list PPSUM( $D_i$ ) of all publishers in  $S_{can}(D_i)$ . We refer to this index as  $IDX_{max}(D_i)$ . Please refer to [15] for the pseudo-code of computing  $S_{can}(D_i)$  and  $IDX_{max}(D_i)$ . Note that a data cube such as SALES is updated in an append-only fashion. When the new sales data of date  $D_i$  are appended to the data cube, we simply compute  $S_{can}(D_i)$  and  $IDX_{max}(D_i)$  based on  $S_{can}(D_{i-1})$  and  $S_{can}(D_0)$ .

We now show how to use  $S_{can}(D_i)$  and  $IDX_{max}(D_i)$  to reduce the list traversals of the bi-directional traversal algorithm, resulting in the *dominant-set oriented algorithm*. We first calculate a set of candidate publishers  $S_r$  that are in  $S_{can}(D_e)$  but not in  $S_{can}(D_s - 1)$ . The publishers in  $S_r$  may or may not be ranked higher than  $IDX_{max}(D_s - 1)$  which is pre-computed. Let  $idx_r$  be the maximum index of the publishers in  $S_r$ . Hence we need to identify the pruning marker  $PM$  which is  $\max(IDX_{max}(D_s - 1), idx_r)$ . Consider the example shown in Fig. 2(b). Given  $AR(2, D_1, D_2)$ ,  $S_r = S_{can}(D_2) - S_{can}(D_0) = \{P_1, P_2\}$ . The  $PM$  for list PPSUM( $D_0$ ) is the maximum value of  $idx_r$  and  $IDX_{max}(D_0)$ .  $idx_r$  in this case is 3 while  $IDX_{max}(D_0)$  is 1. Hence  $PM = 3$ . Thus publisher  $P_4$  can be pruned from the search space.

The rest of the dominant-set oriented algorithm is the same as the bi-directional traversal algorithm except that the starting point of traversing PPSUM( $D_s - 1$ ) is from the pruning marker  $PM$ . Again, due to the space limit, please refer to [15] for the pseudo-code of the algorithm. Since the dominant-set oriented algorithm prunes the search space in PPSUM( $D_s - 1$ ) by applying a pruning marker, it will always outperform the bi-directional traversal algorithm, especially when there is a dominant publisher set.

## 7 Experiments

We conducted extensive experiments over both synthetic and real datasets to evaluate our proposed techniques. The experimental results validated our assumptions regarding the characteristics of datasets. Due to the lack of space, we only present partial experimental results over the real data sets. Please refer to [15] for more performance evaluation.



The real clicks datasets are from CJ.com, an online advertisement tracking company. They are for a larger number of advertisers where the number of publishers for each advertiser ranges between 4,000 and 5,000. The maximum number of publishers is up to 100,000 for some advertisers, however for confidentiality, we were only supplied with the datasets restricted to about 4,000 to 5,000 publishers. Each clicks dataset contains the daily clicks of all publishers for about 180 days.

The experiments were conducted on a Pentium IV 1.6GHz PC with 256MB RAM and 30GB hard disk. We executed two sets of aggregation ranking queries: a *uniform query set* and a *biased query set*, each of them with 1,000 queries. The uniform query set contains queries whose ranges are uniformly generated along the DATE dimension, The biased query set contains queries which are generated to model real user query patterns. The details of generating such a biased query set can be found in [15]. The comparison of the different techniques is based on the average query time in milliseconds.

We conducted experiments over a large number of real clicks datasets of different advertisers to examine how the value of  $k$  affects query cost. The experiments exhibited similar results. Thus, here we only present the experimental results for an advertiser with 4,000 publishers and  $k_{max} = 50$ . Fig. 3(a) and Fig. 3(b) show the experimental results over a uniform query set and a biased query set respectively. We observe that the dominant-set oriented algorithm outperforms both the complete scan algorithm and the bi-directional traversal algorithm. The value of  $k$  does not affect the complete scan approach, since the value of  $k$  does not have any impact on this algorithm assuming that the output time can be ignored. The average query cost of the two other techniques tends to increase slightly when the value of  $k$  increases, since the number of publishers in  $L_s \cap L_e$  becomes larger and therefore results in a larger number of publishers in  $L_s \cup L_e$ . We also notice that the bi-directional traversal algorithm performs worse than the complete scan algorithm. This is because the real clicks datasets demonstrate to have a set of dominant publishers. Also this does not contradict the theoretical analysis as given in [15], which states that the bi-directional traversal algorithm at most needs to process all publishers and has linear performance in the worst case. Due to the dominant publishers, when using the bi-directional traversal algorithm, the number of publishers

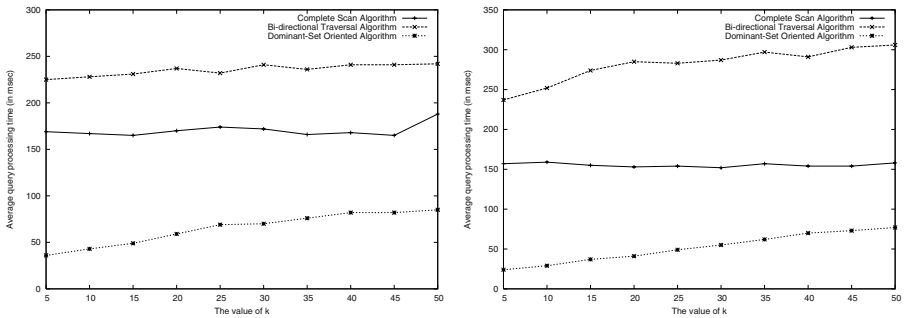


Fig. 3. Performance evaluation over real data set

in  $L_s \cup L_e$  reaches  $n$  ( $n$  is the total number of publishers). Based on the Algorithm 1, in order to compute the total sales for each publisher in  $L_s \cup L_e$ , we need to randomly access the prefix sums of the sales for publishers that are in  $L_s$  but not in  $L_e$  or vice versa. Since the number of publishers in  $L_s \cup L_e$  is almost  $n$ , we need nearly  $n$  random accesses, which results in expensive disk I/O cost. However, in the complete scan algorithm, lists  $\text{PPSUM}(D_s - 1)$  and  $\text{PPSUM}(D_e)$  are always loaded into main memory sequentially thus taking advantage of the fast sequential access property of disks. As a result, the bi-directional traversal algorithm has worse performance than the complete scan algorithm even though they process almost the same number of publishers.

## 8 Conclusion

In this paper, we formalized the notion of aggregation ranking for data warehouse applications. Aggregation ranking queries are critical in OLAP applications for decision makers in the sense that they provide ordered aggregation information. We have proposed a progression of three different algorithms to handle aggregation ranking queries. Our final algorithm, the dominant-set oriented algorithm, is efficient and realistic, since it exploits the pre-computed cumulative information and the bi-directional traversal of lists while restricting the traversal to a small superset of the actual dominant set which is exhibited in real datasets. In general, with increasing reliance on online support for interactive analysis, there is a need to provide query processing support for complex aggregation queries in large data warehouses where sub-query results are correlated on a variety of metrics. Our future work will involve identifying such types of queries and developing database technologies for efficiently processing such queries. Furthermore, our proposed techniques can be generalized to handle aggregation ranking queries over high dimensional data cubes.

## References

1. Brian Babcock and Chris Olston. Distributed top-k monitoring. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 563–574, 2003.
2. N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. on Database Systems*, 27(2):153–187, 2002.
3. N. Bruno, L. Gravano, and A. Marian. Evaluating top-k queries over web accessible databases. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, pages 369–380, 2002.
4. K. C. Chang and S. Hwang. Minimal probing: Supporting expensive predicates for top-k queries. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 346–357, 2002.
5. M. Charikar, K. Chen, and M. Farach-Colton. Approximate frequency counts over data streams. In *Proc. of 29th Int. Colloq. on Automata, Languages and Programming*, pages 693–703, 2002.
6. C.Ho, R.Agrawal, N.Megiddo, and R.Srikant. Range queries in olap data cubes. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, pages 73–88, 1997.
7. D. Donjerkovic and R. Ramakrishnan. Probabilistic optimization of top N queries. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 411–422, 1999.
8. Ronald Fagin. Combining fuzzy information from multiple systems. In *Proc. of Symp. on Principles of Database Systems (PODS)*, pages 216–226, 1996.

9. L. Golab, D. DeHaan, E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro. Identifying frequent items in sliding windows over on-line packet streams. In *Proc. of the conference on Internet measurement conferenc*, pages 173–178, 2003.
10. I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Joining ranked inputs in practice. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 950–961, 2002.
11. I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 754–765, 2003.
12. J.Gray, A.Bosworth, A.Layman, and H.Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tabs and sub-totals. In *Proc. of Int. Conf. on Data Engeering(ICDE)*, pages 152–159, 1996.
13. S. Y. Lee, T. W. Ling, and H.-G. Li. Hierarchical compact cube for range-max queries. In *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*, pages 232–241, 2000.
14. C. Li, K. C.-C. Chang, I.F. Ilyas, and S. Song. Ranksql: Query algebra and opatimization for relational topk queries. In *Proc. of Int. Conf. on Management of Data (SIGMOD)*, 2005.
15. H.-G. Li, H. Yu, D. Agrawal, and A. El Abbadi. Ranking aggregates. Technical Report 2004-07, University of California at Santa Barbara, <http://www.cs.ucsb.edu/research/trcs/docs/2004-07.pdf>, 2004.

# On Efficient Storing and Processing of Long Aggregate Lists

Marcin Gorawski and Rafal Malczok

Silesian University of Technology,  
Institute of Computer Science,  
Akademicka 16, 44-100 Gliwice, Poland  
{Marcin.Gorawski, Rafal.Malczok}@polsl.pl

**Abstract.** In this paper we present a solution called Materialized Aggregate List designed for the efficient storing and processing of long aggregate lists. An aggregate list contains aggregates, calculated from the data stored in the database. In our approach, once created, the aggregates are materialized for further use. The list structure contains a table divided into pages. We present three different page-filling algorithms used when the list is browsed. We present test results and we use them for estimating the best combination of the configuration parameters: number of pages, size of a single page and number of available database connections. The Materialized Aggregate List can be applied on every aggregation level in various indexing structures, such as, an aR-tree.

## 1 Introduction

Query evaluation time in relational data warehouse implementations can be improved by applying proper indexing and materialization techniques. View materialization consists of first processing and then storing partial aggregates, which later allows the query evaluation cost to be minimized, performed with respect to a given load and disk space limitation [9]. In [5] the authors for the first time use the spatial network for storing the relations between aggregated views. In [1,4] materialization is characterized by workload and disk space limitation. Indices can be created on every materialized view. In order to reduce problem complexity, materialization and indexing are often applied separately. For a given space limitation the optimal indexing schema is chosen after defining the set of views to be materialized [2]. In [6] the authors proposed a set of heuristic criteria for choosing the views and indices for data warehouses. They also addressed the problem of space balancing but did not formulate any useful conclusions. [8] presents a comparative evaluation of benefits resulting from applying views materialization and data indexing in data warehouses focusing on query properties. Next, a heuristic evaluation method was proposed for a given workload and global disk space limitation.

In this paper we present a new approach to storing and processing of long aggregate lists. In our approach we materialize the calculated values (query results), but we divide the data set into smaller sets that we call pages. Our paper

is organized as follows: section 2 briefly describes the motivation for our work. In section 3 all the specification and configuration aspects are presented. Section 4 describes all the most interesting details of the proposed solution, and in section 5 we present the current state of the art. In section 6 we present test results. Finally, section 7 concludes the paper.

## 2 Motivation

We are working in the field of spatial data warehousing. Our system (Distributed Spatial Data Warehouse – DSDW) presented in [3] is a data warehouse gathering and processing huge amounts of telemetric information generated by the telemetric system of integrated meter readings. The readings of water, gas and energy meters are sent via radio through the collection nodes to the telemetric server. A single reading sent from a meter to the server contains a timestamp, a meter identifier, and the reading values. Periodically the extraction system loads the data to the database of our warehouse.

In our current research we are trying to find the weakest points of our solution. After different test series (with variations of aggregation periods, numbers of telemetric objects etc.) we found that the most crucial problem is to create and manage long aggregate lists. The aggregate list is a list of meter reading values aggregated according to appropriate time windows. A time window is the amount of time in which we want to investigate the utility consumption. The aggregator is comprised of the timestamp and aggregated values.

When we want to analyze utility consumption we have to investigate consumption history. That is when the aggregate lists are useful.

In the system presented in [3] aggregate lists are used in the indexing structure that is a modification of an aR-Tree [7]. Every index node encompasses some part of the region where the meters are located and has as many aggregate lists as types of meters featured in its region. If there are several meters of the same type, the aggregate lists of the meters are merged (aggregated) into one list of the parent node.

The aggregate lists are stored in the main computer memory. Memory overflow problems may occur when one wants to analyze long aggregation periods for many utilities meters. If we take into consideration the fact that the meter readings should be analyzed every thirty minutes, simple calculations reveal that the aggregate list grows very quickly with the extension of an aggregation period. For instance, for single energy meter an aggregate list for one year has  $365 \cdot 48 = 17520$  elements. Each of the aggregators creating the list stores a few values, so the memory consumption is high. In order to prevent memory overflows we designed a memory managing algorithm applied in the system presented in [3]. The mechanism defines a memory limit when the system starts. The limit is always checked before some new aggregate list is created. If upon being loaded a new list threatens to exceed a limit, the mechanism searches for a less frequently read node in the indexing structure and removes its aggregate lists from the memory, providing space for the new lists. The mechanism performs well when

system uptime is not long. The creation and removal of aggregate list produces memory fragmentation that results in memory overflow errors, even though the memory limit had not been exceeded. Hence we decided to search for a new approach to storing and processing aggregate lists with no length limitations. Our main objectives were: the solution must be efficient and scalable, applicable in indexing structures such as aR-tree and easy to use. We named the solution a Materialized Aggregate List (MAL).

### 3 Specification

The main idea of the proposed solution is to provide a user with a simple interface based on the standard Java list mechanism – a set of two functions: *hasNext()* and *next()* which permits the convenient browsing of the list contents. Our purpose was to create a list that could be used as a tool for mining data from the database as well as a component of indexing structure nodes (fig. 1). Below we present an example showing how the list can be used in the program code.

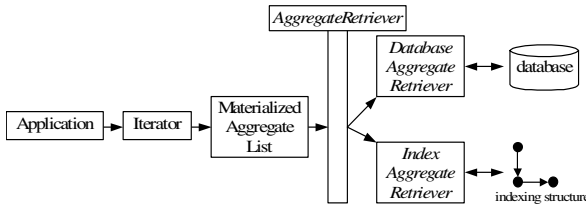


Fig. 1. MAL idea – provide a solution based on a well-known standard

```

(1) MALList list = new MALList(categ, ob, dbConn);
(2) Iterator iterator = list.iterator(startDate);
(3) while (iterator.hasNext()){
(4)   Aggregator a = (Aggregator)iterator.next();
(5)   /* use theaggregator */
(6)   /* time condition breaking the iteration */
(7) }
(8) list.close();
  
```

In the first line we see how the list object is constructed. The constructor parameters are: the category (defines list type), an identifiable object (spatial telemetric object or indexing structure node) and a database connector.

The second line creates the iterator that allows list browsing. In the standard Java implementation the iterator function has no parameter. In the case of MAL there is one parameter defining the timestamp of the first aggregator returned by the *next()* function call.

Lines 3-7 contain instructions known from the standard Java solution. First (line 3) it checks the availability of the next element in the list; the element is retrieved (line 4) and some operations using this element are performed (line

5). A time condition can be put in the next line, breaking the iteration. The condition may be applied if it is not necessary to browse the whole list.

Line 8 closes the list. No new iterators can be created, the list waits for all running threads to complete.

## 4 MAL Details

As mentioned before, our main intention when designing the MAL was to build a solution free of memory overflows which would allow aggregate list handling with no length limitations. We applied the following approach: every list iterator consists of a table divided into pages. When an iterator is created some of the pages are filled with aggregators (which pages and how many is defined by the applied page-filling algorithm, see description below). The next pages are filled (the aggregators are retrieved from the iterator table), while the list is being browsed. After the whole page is read, it is refilled with new data. The solution also uses an aggregates materialization mechanism that strongly speeds up the aggregates retrieval. The most crucial configuration aspects are: the number of pages, the size of a single page, the number of available database connections and the pages-filling algorithm.

The actual list operation begins when a new iterator is created (*iterator()* function call). A new table is created and two values are calculated:

- border date. The border date is used for managing the materialized data. The border date is calculated by repeatedly adding to the install date (defined in category block of the configuration file) a width of aggregation window multiplied by the size of the table page. The date is equal to the timestamp of the first aggregator in the page.
- starting index. In the case that starting date given as a parameter in the *iterator()* function call is different from the calculated border date, the iterator index is adjusted so that a the first *next()* function call returns the aggregator with the timestamp nearest to the given starting date.

### 4.1 Page-Filling Algorithms

As a new iterator is constructed some of its table pages are filled with aggregators. Which pages and how many of them depends on the used page-filling algorithm. All the algorithms create the page-filling threads that operate according to the following steps:

1. Check whether some other thread filling a page with an identical border date is currently running. If yes, register in the set of waiting threads.
2. Get a database connection from the connection pool.
3. Check if the required aggregates were previously calculated and materialized. If yes, restore the data and go to 5.
4. Create the aggregate list. Materialize the list.
5. Release the database connection.

6. Browse the set of waiting threads for threads with the specified border date. Transfer the data and notify them.

In the subsections below we present three different page-filling algorithms.

**Algorithm SPARE.** Two first pages of the table are filled when a new iterator is being created and the SPARE algorithm is used as a page-filling algorithm. Then, during the list browsing, the algorithm checks in the *next()* function if the current page (let's mark it  $n$ ) is exhausted. If the last aggregator from the  $n$  page was retrieved, the algorithm calls the page-filling function to fill the  $n + 2$  page while the main thread retrieves the aggregates from the  $n + 1$  page. One page is always kept as a "reserve", being a spare page. This algorithm brings almost no overhead – only one page is filled in advance. If the page size is set appropriately so that the page-filling and page-consuming times are similar, the usage of this algorithm should result in fluent and efficient list browsing.

**Algorithm RENEW.** When the RENEW algorithm is used, all the pages are filled during creation of the new iterator. Then, as the aggregates are retrieved from the page, the algorithm checks if the retrieved aggregator is the last from the current page (let's mark it  $n$ ). If the condition is true, the algorithm calls the page-filling function to refill the  $n$  page while the main thread explores the  $n + 1$  page. Each time a page is exhausted it is refilled (renewed) immediately. One may want to use this algorithm when the page consuming time is very short (for instance the aggregators are used only for drawing a chart) and the list browsing should be fast. On the other hand, all the pages are kept valid all the time, so there is a significant overhead; if the user wants to browse the aggregates from a short time period but the MAL is configured so that the iterators have many big pages – all the pages are filled but the user does not use all of the created aggregates.

**Algorithm TRIGG.** During new iterator creation by means of the TRIGG algorithm, only the first page is filled. When during  $n$  page browsing the one before last aggregator is retrieved from the page the TRIGG algorithm calls the page-filling function to fill the  $n + 1$  page. No pages are filled in advance. Retrieving the next to last aggregator from the  $n$  page triggers filling the  $n + 1$  page. The usage of this algorithm brings no overhead. Only the necessary pages are filled. But if the page consumption time is short the list-browsing thread may be frequently stopped because the required page is not completely filled.

## 4.2 Connection Pool

A very important aspect of the Materialized Aggregate List operation is database access. The page-filling threads use the database connection for creating an aggregate list and for list materialization and restoring. The connection can be used by only one thread at a time. The connection retrieving operation may cause some threads to stop when the number of concurrently running threads is greater than the number of available connections. To optimize connection management we decided to use the concept of connection pool (generally: resource



pool) and connection factory (generally: resource factory). The pool parameter is the maximal number of connections that can be obtained from the pool. After creating, the pool does not contain any connections. During application operation, any thread that requires a database connection calls a pool method for retrieving a connection. Depending on the pool state the following operations are performed:

- if the pool contains a free connection, the connection is assigned to the calling thread,
- if the pool does not contain a free connection, but the connections limit is not exceeded, a new connection is created by means of the connection factory and assigned to the calling thread
- if the pool does not contain a free connection and creating a new connection would cause the connections limit to exceed, the calling thread is stopped until some connection is returned to the pool or the pool is destroyed.

When a thread completes the operations requiring database connection, the connection is returned to the resource pool. If some threads are waiting for a connection, one of them will be assigned a connection and notified.

### 4.3 Materialization

In the presented operation of the page-filling function, points (3) and (4) mention a concept of materialization. We introduced the materialization mechanism in the DSDW system presented in [3] and the tests revealed the mechanism extreme efficiency. The idea is to store once calculated aggregators as binary data in the database, using the BLOB table column. In the current approach we use a table with three columns storing the following values: the object identifier (telemetric object or indexing structure node), page border date and aggregators in binary form. The page materialization mechanism operates identically for each page-filling algorithm.

## 5 State of Art and Future Plans

After finishing work on theoretical concepts we started implementation of our solution. The current state of the art contains a full implementation of the database iterator (the iterator for retrieving aggregates from a database) and all three page-filling algorithms. The list operation is convergent with the description presented in section 3. The iterator retrieving aggregates from the database can automatically process new data added by the extraction process. If some page was materialized but it is not complete (not all necessary data was found in the database when it was being filled), then the page-filling thread starts exploring the database from the point where the data was not available. The aggregates retrieving finishes if there is no more available data, then the *hasNext()* function call returns false.

We are nearing completion of the work on the MAL iterator, which permits us to apply the solution in indexing structures (such as aR-Tree). The applied page-filling algorithm is very similar to the TRIGG algorithm for the database iterator.

The data warehouse structure described in [3] applies distributed processing. We also suppose that in this aspect introducing the MAL to our system will bring benefits in efficiency. The current approach to sending complete aggregate lists as a partial result from a server to a client results in high, single client module load. When we divide the server response into MAL pages, the data transfer and the overall system operation will presumably be more fluent.

## 6 Test Results

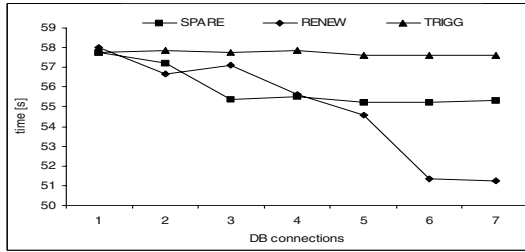
This section contains a description of the tests performed with the current implementation of the presented solution. The tests were executed on a machine equipped with Pentium IV 2.8 GHz and 512 MB RAM. The software environment was Windows XP Professional, Java Sun 1.5 and Oracle 9i. The tests were performed for all three page-filling algorithms. Each of the algorithms was applied in the iterator used for retrieving aggregates from the database for 3, 6, 9, and 12 months. The aggregates were created with a time window of 30 minutes. The created aggregates were not used in the test program; the program only sequentially browsed the list. Aggregates browsing was performed twice: during the first run the list has no access to the materialized data, and during the second run a full set of materialized data was available. The MAL parameters, page number, page size and the number of available database connections, had the following values:

- page size: 48 (1 day), 240 (5 days), 336 (7 days), 672 (14 days), 1008 (21 days), 1488 (31 days – 1 month), 2160 (46 days – 1.5 month), 2976 ( 62 days – 2 months) and 4464 (93 days – 3 months),
- page number:  $2 \div 10$ .
- number of database connections:  $1 \div pageNumber + 1$

Our goal was to find the best combination of the MAL parameters: the page-filling algorithm, number of pages, size of a single page and number of available database connections. The choice criterion consisted of two aspects: the efficiency measured as a time of completing the list-browsing task and memory complexity (amount of the memory consumed by the iterator table).

### 6.1 Page Size and Page Number

We first analyze the results of completing the list-browsing task during the first run (no materialized data available) focusing on the influence of the page size parameter. We investigated the influence for various numbers of pages and for all three algorithms always setting the number of available database connections to 1. We observe that for all three algorithms the influence is very similar; graphs of



**Fig. 2.** Operation of the SPARE algorithm for retrieving aggregates for 6 months

the relations are very convergent. Figure 2 presents a graph showing the results obtained for the SPARE algorithm for the aggregation period of 6 months. The list browsing times for small pages are very diverse. For the presented results the times for a page of size 48 vary from 30 to 160 seconds depending on the amount of pages. MAL operation for a page of size 240 is much more stable; the differences resulting from the different number of pages do not exceed 25 seconds. In graph we observe that for pages greater or equal 672 the list browsing time does not significantly depend on the number of pages. We must notice that the page size strongly influences the amount of memory consumed by the iterator. Hence, considering the fact that further increasing the page size brings almost no time benefit, we chose the page size 672 as the most optimal.

As next, we analyzed the influence of the combination of two parameters: number of pages and number of available database connections on the MAL efficiency. We performed the test for 1 to  $number\_of\_pages + 1$  available connections because in some particular cases the MAL instance also utilizes a database connection. Again, we must notice, that number of pages influences the amount of consumed memory as well as the CPU workload (in the worst case the number of pages equals the number of concurrently running threads). Analyzing test results we concluded the following: the most optimal benefit/cost ratio is when the list is configured to work with  $4 \div 6$  pages and the connection pool contains as many connections as there are pages.

## 6.2 Page-Filling Algorithm

After choosing the optimal parameters, we compared the time efficiency of the page-filling algorithms. Figure 3 shows a graph comparing efficiency of the algorithms for browsing the list of aggregates for 12 months. The list was configured to use 6 pages, each of size 672. The obtained results are strictly coherent with the theoretical assumptions of the page-filling algorithms. When only one database connection is available there is no time difference in the operation of the algorithms. But along with increasing the number of available connections the SPARE and the RENEW algorithms show better efficiency while the TRIGG algorithm efficiency remains unchanged. The TRIGG algorithm fills only one page at a time; it uses only one database connection. As a result, increasing the

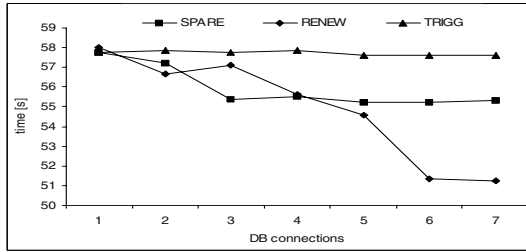


Fig. 3. Comparison of the page-filling algorithms

number of available connections brings no time profit. The STEPS algorithm launches at maximum 2 threads concurrently, utilizing at most 3 database connections what is clearly seen in the graph. And finally, the RENEW algorithm fills all the pages concurrently, utilizing all the available connections. It improves its efficiency each time the number of database connections increases. We chose this algorithm as the most efficient one.

Therefore, to summarize the parameters selection we can state that the MAL works efficiently for the following configuration: the RENEW algorithm, number of pages  $4 \div 6$ , size of a single page 672, number of available database connections equals number of pages.

### 6.3 Materialization

The aspect last investigated was materialization influence on system efficiency. The results interpretation reveals that materialization strongly improves system efficiency. In figure 4 there is a graph showing the MAL operation for the TRIGG algorithm for various number of pages of sizes 672 and 1488 and with one database connection. As the first run we marked the list operation with no materialized data, and as a second run we marked the operation with the full set of materialized data. In both page size variants the benefit of materialization is very similar, and upon analyzing the charts, we can state that using the materi-

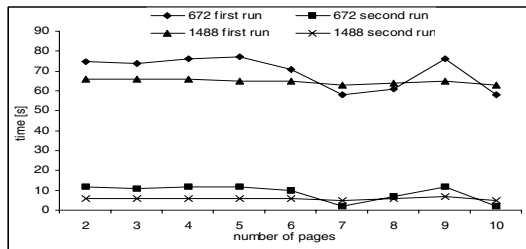


Fig. 4. Comparison the first and second run for the TRIGG algorithm with pages containing 672 and 1488 aggregators

alized data the list operates from 5 to 8 times faster than when no materialized is used. A similar situation can be observed for all the page size and page number parameter combinations.

## 7 Conclusions

In this paper we presented the Materialized Aggregate List (MAL). The MAL is a data structure for storing long aggregate lists. The list can be applied as a component of indexing structure nodes in indexes like an aR-Tree. The aggregators stored in the list can be retrieved from both the database and from other levels of an indexing structure. In our solution we applied the idea of aggregates materialization. The materialization has a very strong, positive influence on list efficiency. We presented the current state of the art, our future plans, and the theoretical and practical details of our solution. The paper additionally describes results of the preliminary tests.

## References

1. Baralis E., Paraboschi S., Teniente E.: Materialized view selection in multidimensional database. In Proc. 23<sup>th</sup> VLDB, pages 156-165, Athens, 1997.
2. Golfarelli M., Rizzi S., Saltarelli E.: Index selection for data warehousing. In. Proc. DMDW, Toronto, 2002.
3. Gorawski M., Malczok R.: Distributed Spatial Data Warehouse Indexed with Virtual Memory Aggregation Tree. 5<sup>th</sup> STDBM\_VLDB'04 Workshop, Toronto 2004
4. Gupta H.: Selection of views to materialize in a data warehouse. In. Proc. ICDT, pages 98-112, 1997.
5. Harinarayan V., Rajaraman A., Ullman J.: Implementing data cubes efficiently. In. Proc. ACM SIGMOD Conf., Montreal, 1996.
6. Labio W.J., Quass D., Adelberg B.: Physical database design for data warehouses. In. Proc. ICDE, pages 277-288, 1997.
7. Papadias D., Kalnis P., Zhang J., Tao Y.: Efficient OLAP Operations in Spatial Data Warehouses. Springer Verlag, LNCS 2001
8. Rizzi S., Saltarelli E.: View Materialization vs. Indexing: Balancing Space Constraints in Data Warehouse Design, CAISE, Austria 2003
9. Theodoratos D., Bouzehoub M.: A general framework for the view selection problem for data warehouse design and evolution. In. Proc. DOLAP, McLean, 2000

# Ad Hoc Star Join Query Processing in Cluster Architectures

Josep Aguilar-Saborit<sup>1</sup>, Victor Muntés-Mulero<sup>1</sup>, Calisto Zuzarte<sup>2</sup>, and Josep-L. Larriba-Pey<sup>1</sup>

<sup>1</sup> Universitat Politècnica de Catalunya, Computer Architecture Department, Jordi Girona 1-3. Campus Nord-UPC, Modul D6, E-08034, Barcelona, Spain  
{jaguilar, vmuntes, larri}@ac.upc.edu

<sup>2</sup> IBM Toronto Lab. 8200 Warden Ave. Markham, ON L6G1C7, Canada  
calisto@ca.ibm.com

**Abstract.** Processing of large amounts of data in data warehouses is increasingly being done in cluster architectures to achieve scalability. In this paper we look into the problem of ad hoc star join query processing in clusters architectures. We propose a new technique, the Star Hash Join (SHJ), which exploits a combination of multiple bit filter strategies in such architectures. SHJ is a generalization of the Pushed Down Bit Filters for clusters. The objectives of the technique are to reduce (i) the amount of data communicated, (ii) the amount of data spilled to disk during the execution of intermediate joins in the query plan, and (iii) amount of memory used by auxiliary data structures such as bit filters.

## 1 Introduction

The use of clusters architectures and in particular those with Symmetric MultiProcessing (SMP) nodes, has become a popular solution to implement massive parallelism. In November 2004, about 60% of the supercomputers in the 'TOP500' list [1] were cluster computers, showing a significant growth in the last few years. Database applications show a high demand on the use of these type of configurations: the amount of data is approaching the petabyte steadily, moving research in data warehousing towards many implementation challenges. SMP configurations by themselves show scalability problems. This is solved by connecting multiple clusters in an SMP configuration through a network communication system, offering high scalability and cost effectiveness. This kind of configuration is also known as a CLUMP architecture.

In order to balance the load, clusters share large volumes of data in such a way that each cluster keeps a portion of the overall database. While the methodology proposed in this paper is applicable to clusters that are not necessarily in an SMP configuration, the CLUMP architecture is commonly used, and thus the configuration chosen for evaluation in this paper. A cluster within an SMP configuration consists of a set of processors that share memory and I/O resources under the control of one copy of the operating system. All the processes access the memory by using high-speed buses or advanced cross-switching technologies that support point-to-point interconnections between processors.

CLUMP architectures speed up query processing by balancing large volumes of data across all clusters in the system. Thus, each cluster can work in parallel with its own part of the overall database. The clustering scheme alleviates the problem of I/O processing for large relations. Moreover, SMP processes scan in parallel the data that may be striped across multiple disks within each cluster. However, with data partitioned across the clusters, data communication becomes important, and makes the optimization of data processing even more relevant to achieve good performance.

**Ad hoc Star Joins and CLUMP architectures.** Data warehouse environments are generally organized according to a multidimensional model with one or more star schemas [11]. Each star schema consists of a very large central Fact table surrounded by multiple dimension tables that are linked to it through primary and foreign key relationships. On-Line Analytical Processing (OLAP) queries are usually complex and ad hoc with high selectivity factors. Queries are not known in advance and have a multi-table join flavour, joining the Fact table with its corresponding dimensions. These queries are also called *ad hoc star join queries* in the literature [10].

The distribution of a database in CLUMP architectures is done by applying a shipping function that decides the host cluster of each record in a table. The shipping function is applied to a column or set of columns (called the *partitioning key*) from the table to be partitioned [15]. When performing a join operation, if the joining key is the same as the partitioning key used for both input tables, then the join is said to be *collocated*: if not, the join is said to be *non-collocated*. Collocated joins can be performed locally within each cluster, while non-collocated joins need for a dynamic re-partitioning of the data. In this case, it is necessary to either broadcast the data usually from the smaller table, or selectively re-partition both relations.

When executing a star join in CLUMP architectures, the query execution plan is the same for all members of the clusters. The main difference between a parallel and a sequential environment is data communication. The minimization of inter-cluster data traffic becomes a priority during star join processing because of the large volumes of data potentially shipped.

**Contributions of the paper.** We propose a technique that saves both intra cluster data processing and inter-cluster communication during ad hoc star join processing. We call our strategy *Star Hash Join* (SHJ) and it is a generalization of Pushed Down Bit Filters (PDBF) [3] for CLUMP architectures. PDBF is a technique used in a shared-everything environment and consists of the filtering of data in advance by pushing down the bit filters [8] to the lower operations of the query execution plan. SHJ generalizes PDBF for clusters architectures using Semi-join reduction, that is primarily for shared nothing environments [6][7].

**Organization of the paper.** We start by explaining our proposal, the Star Hash Join in the context of existing work. In Sections 3 and 4, we explain the evaluation setup environment and analyze our proposal. We wrap up the related work in Section 5, and in Section 6 we draw some conclusions.

## 2 Star Hash Join (SHJ)

In this section we explain our proposal, the Star Hash Join. We will go through some basic concepts before introducing the algorithm.

**Hybrid Hash Join.** The Hybrid Hash Join [13] is a hash-based join algorithm commonly used to perform hash join operations in DBMSs such as the IBM<sup>®</sup> DB2 Universal Database<sup>TM</sup> (DB2 UDB), Microsoft<sup>®</sup> SQL-Server or the Oracle<sup>®</sup> database products. It consists of two phases: (1) a build phase, where a hash table is built with tuples, usually from the smaller input relation, and (2) a probe phase, where tuples from the larger relation are used to probe the hash table looking for matches. If the hash table created during the build phase does not fit in memory, then hashing [8] is used to partition both input relations in the same way such that each partition from the smaller relation can fit in memory, and the join can be performed on each pair of corresponding partitions.

The Hybrid Hash Join may use bit filters [8] to speed up its execution. Bit filters are created during the build phase of a join. By applying hashing, the value of the joining key for each record from the build relation is mapped into the bit filter, and its corresponding bit is set to 1. Then, the bit filter is checked using the same hashing function for the joining key of each record processed during the probe phase. If the corresponding bit was not set during the build phase, then the tuple can be filtered out. The main goal of the use of the bit filter is to avoid spilling tuples to disk during the probe phase, saving I/O.

When performing a parallel non-collocated Hybrid Hash Join the build relation or the probe relation, or both, may need to be re-partitioned. A common join plan strategy is to selectively repartition the probe relation and send it to a specific target cluster in which the build relation is already partitioned on the join key. Another common join plan strategy is to broadcast the build relation to all the clusters that contain the probe relation. We focus on the former case.

**Star Hash Join Schema.** A star join query with  $n$  dimension tables and a Fact table, may be performed through  $n$  Hybrid Hash Join operations as a left-deep tree-shaped query plan. Figure 1 shows the shape of a Star Hash Join schema for a system configuration with  $k$  clusters. We use the following definitions :

- $D_{ij}$  is the part of the  $i^{th}$  dimension table stored in cluster  $j$ , and  $BF_{ij}$  is the bit filter created during its build phase.
- $Fact_j$  is the portion of the Fact table stored in cluster  $j$ .
- RO is the Re-partitioning Operator responsible for selectively sending data to the rest of the computational clusters, as well as receiving data from each of those clusters.

Typically, in a star schema, the dimension tables are partitioned by their primary keys, and the Fact table is partitioned by one of its foreign keys. Hence, only one join between one dimension table and the Fact table may be collocated. The rest of the joins will be non-collocated, and will need the presence of an RO during the probe phase of the left-deep tree. Note that if the database system supports it, and if storage space, administration overhead, and synchronization



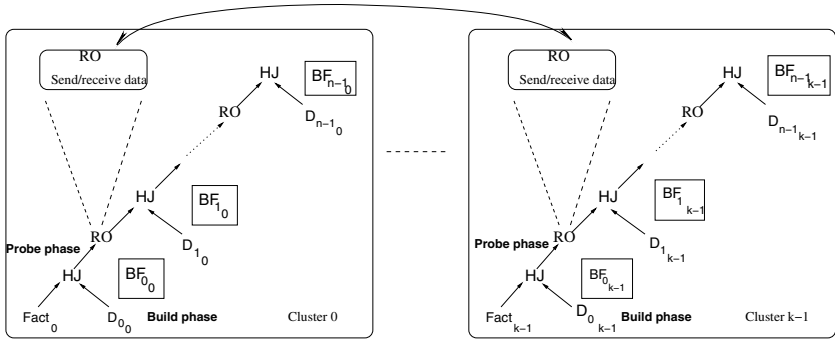


Fig. 1. Star hash join schema for  $k$  clusters

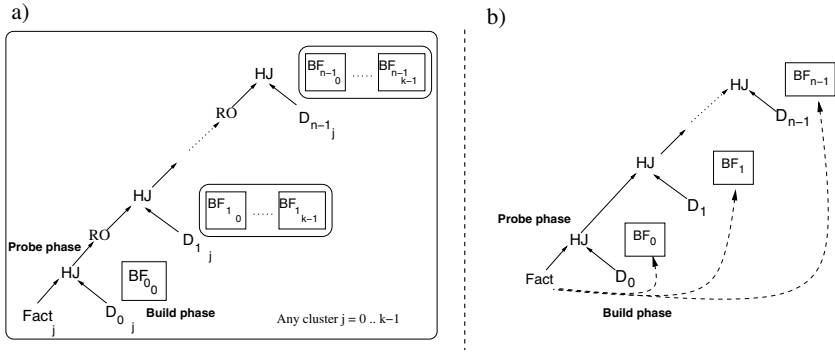
overhead, are not important considerations, the entire dimension tables can be replicated on each cluster before the queries are run. Broadcasting records from these tables in this case need not be done during query processing. In this paper, it is assumed that dimension tables are not replicated on each cluster because this may not be feasible.

## 2.1 The Algorithm

Our approach generalizes the use of Pushed Down Bit Filters (PDBF) for a faster execution of ad hoc star join queries in CLUMP architectures. The Star Hash Join algorithm makes use of Semi-join reduction [7][6] to do such generalization.

The use of Semi-join reduction in clusters has been called Remote Bit Filters Broadcasted ( $RBF_B$ ) [5].  $RBF_B$  broadcasts the bit filters created locally during the build phase to all the clusters involved in the processing of a non-collocated Hybrid Hash Join. Hence, before a record is sent to a remote cluster during the probe phase, the record is tested against the bit filter of the target cluster, and is discarded before being sent if possible.  $RBF_B$  is graphically explained in Figure 2.a. For a given cluster  $j$ , a hash join between one dimension ( $D_{i_j}$ ) and the Fact table ( $Fact_j$ ), will keep bit filters  $BF_{0_j}, BF_{1_j}, ..BF_{n-1_j}$  in its local memory, one for each of the  $n$  dimension tables.

Pushed Down Bit Filters are aimed at saving intra-data processing within shared-everything environments [3]. PDBF uses the bit filters generated in the upper operations of the query plan to filter out tuples in the leaf operations. This saves the processing of data and I/O of the intermediate results for the join operations in between. PDBF is graphically shown in Figure 2.b. Each record scanned from the Fact table is tested against each of the  $n$  bit filters. If any one of the bit filters has the associated entry of the current record set to 0, then the given record can be discarded. PDBF cannot be used across clusters architectures, because it is not possible, using only the local bit filters, to filter out data that has to be transmitted to remote clusters.

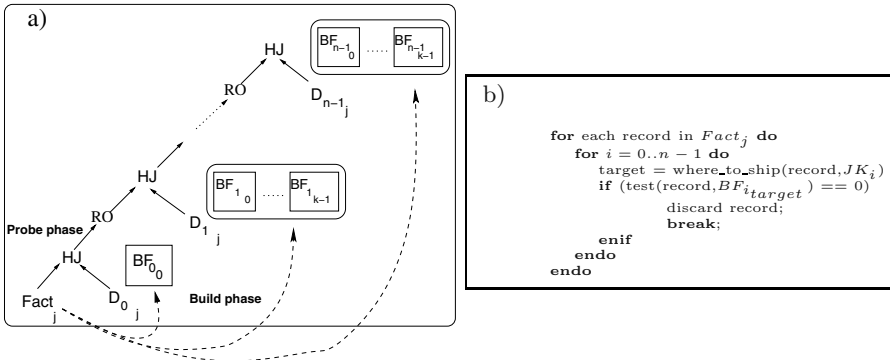


**Fig. 2.** a) Semi-join reduction ( $RBF_B$ ) b) Pushed Down Bit Filters ( $PDBF$ )

**Star Hash Join (SHJ).** The strategy followed by SHJ is graphically explained in Figure 3.a.  $RBF_B$  is applied in order to keep copies of all the bit filters for the star join query in every single cluster. On the other hand,  $PDBF$  allows the lower operations of the query execution plan to have access to all the bit filters of the query. This way, every record of the Fact table is checked against the bit filters of its target cluster, and a record is only transmitted if it has a potential joining record in the remote destination cluster. Thus, SHJ extends the use of  $PDBF$  to clusters. Moreover, it gets a significant major reduction in terms of data communication than just the Semi-join reduction executed alone.

We show the algorithmic version of SHJ in Figure 3.b. Given cluster  $j$  and  $n$  joins between dimension tables ( $D_{i_j}, i = 0..n-1$ ) and the Fact table ( $Fact_j$ ), we name the joining keys  $JK_i$ . Thus, each record being scanned from table  $Fact_j$  is processed as follows:

1. For each join in the query, we apply the shipping function used to distribute data to the joining key (where\_to\_ship( $JK_i$ )). This way, we figure out the target cluster of the current record. Joining keys in a star join are usually



**Fig. 3.** a) Star Hash Join b) The algorithm

primary keys, which are also the partitioning keys, in the dimension tables. Hence, the joining key  $JK_i$  stored in each record from the Fact table, is the only information needed to determine where it has to be shipped.

- Then, the record being processed is tested against the bit filter of the target cluster  $BF_{i_{target}}$ . If it returns zero, the record can be discarded, otherwise it has to be processed.

### 3 Evaluation Set Up

We perform our evaluation analyzing an environment similar to that used for the TPC-H benchmark [2]. We simulate and analyze the execution of one TPC-H like query over five clusters in an SMP configuration, each server with 32 processors sharing 256GB of main memory and 512 disks. A similar configuration has been used by IBM in a 10TB TPC-H benchmark [2].

We assume a 10TB TPC-H database, partitioned across the system with hash partitioning. For our analysis, we have used a query based on TPC-H query 9 (see Figure 4). In Figure 4.a, we show the execution plan of the part of the query that executes the star join, which has the shape of the Star Hash Join schema explained in this paper. The memory available to each hash join is 1.3 GB, and the SMP degree is 32, meaning that the 32 processors will work in parallel sharing resources within each cluster.

The selectivity of the dimension tables will vary depending on the values of  $x, y, z$  and  $w$  shown in the query of Figure 4. In Figure 4.b we show the different selectivity sets applied to the dimension tables that we use in order to analyze a wide range of situations for the different techniques being compared. Bit filters are used in all hash joins no matter what the selectivity is of the build relation. The default fraction of false positives [8]( $F_p$ ) for any bit filter is set to  $F_p = 0.05$ .

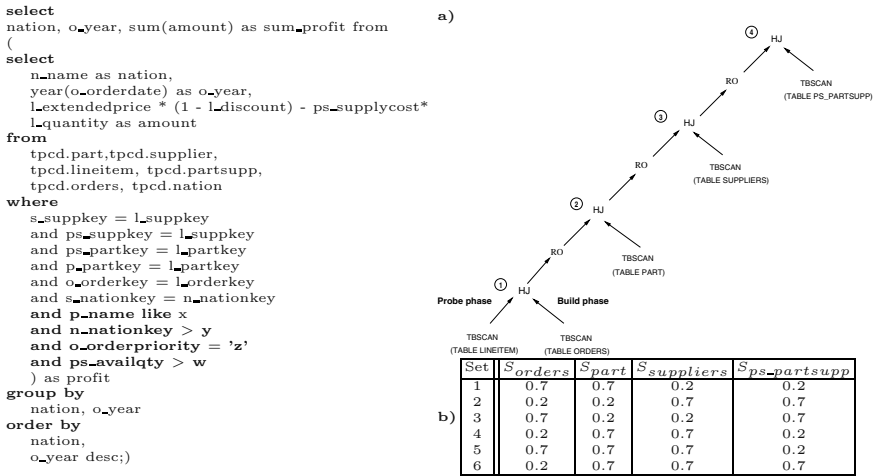


Fig. 4. a) Execution plan b) Sets of selectivities  $S_R$  is the Selectivity of relation R

## 4 Analysis

We perform a detailed analysis based on the mathematical models carefully explained in the extended version of this paper [4]. We compare the following strategies: (i) a baseline where only local bit filters are generated, (ii) Remote Bit Filters Broadcasted (*RBF<sub>B</sub>*) and (iii) the Star Hash Join (SHJ). Pushed Down Bit Filters (PDBF) alone is not applicable in CLUMP architectures, as explained earlier. We show results for the benefit obtained by the techniques compared to the baseline strategy unless otherwise specified. Also, the plots show along their horizontal axis, results for sets 1 to 6, as shown in the table of Figure 4.b.

### Communication and I/O-Join Processing

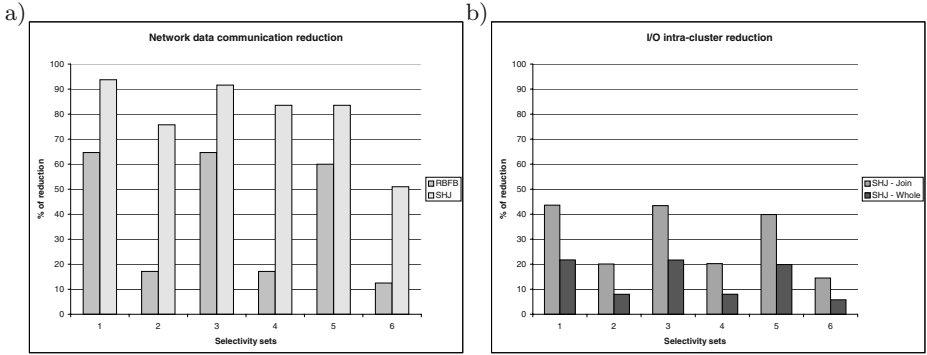
Figure 5 shows the percentage of data communication reduction and that of I/O reduction during join processing. The trends in both plots are quite similar:

- **Data communication.** SHJ always gets at least a 60% improvement, over the baseline. In the best case a 92% improvement over the baseline is achieved. The higher the selectivity of the whole query, the larger the improvement obtained with SHJ, as shown by sets 1, 3, 4 and 5. Also, the effect of PDBF causes SHJ to further reduce inter-cluster communication, with improvements over Semi-join reduction (*RBF<sub>B</sub>*) ranging from 25% to 65%.
- **I/O.** Figure 5.b distinguishes between the I/O incurred by the whole query (SHJ-Whole) and that excluding the scan over the Fact table (SHJ-join). SHJ saves a significant amount of I/O during query processing even in the case when we account for the I/O incurred by the Fact table.

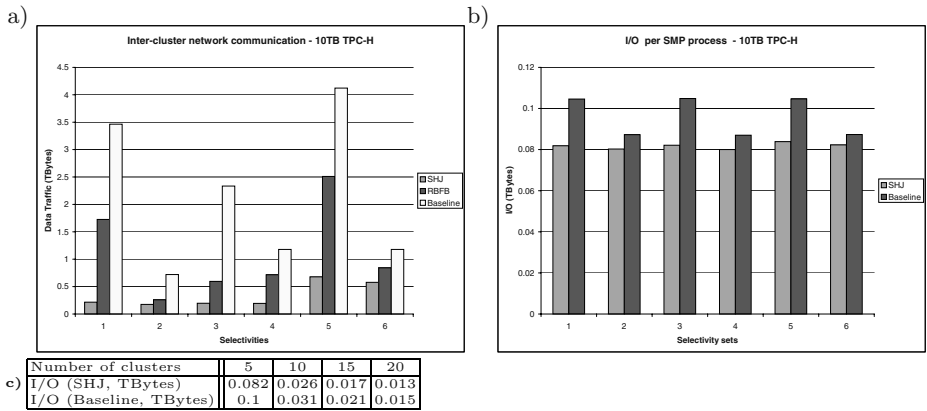
We achieve the best benefits when scan selectivities are low in the dimensions of the upper non-collocated joins, as in sets 1, 3 and 5. That is predictable as we are reducing a considerable amount of data from the Fact table in the leaves of the plan. On the other hand, when the lowest selectivity is placed in the dimension table of the collocated join, as in sets 2, 4 and 6, we get less benefit over the baseline. This is because most of the records from the Fact table are purged in the lowermost join. In this case the star join processing is less costly even in the baseline execution.

### Trade-Off. Communication vs. I/O

Figure 6 shows the amount of communication and I/O in TB modelled during the processing of a star join. We assume that data is stripped across disks within each cluster, and every SMP process can read and write data in parallel with no overlapping. Hence, I/O is calculated for each of the 32 processes running in each cluster. We can see that communication in CLUMP architectures does not scale in the same way as I/O does. In all the cases we get 20 times more network communication than I/O. Figure 6.c shows, for the selectivity set number 1, how the I/O scales as more clusters of the same type are added. While I/O is reduced significantly, communication, being the single resource shared by all clusters in the network, remains the same.



**Fig. 5.** a) Network data comm. reduction b) SHJ-Join: I/O reduction during join processing SHJ-Whole : I/O overall reduction



**Fig. 6.** a) Network data comm b) Total I/O c) I/O scalability

## Memory Resources

Figure 7.a shows the amount of memory in MB required by SHJ for each selectivity set. We can see that SHJ needs in this particular case, at most, 1.8 GB of main memory to keep all bit filters of the system with a fraction of false positives [8],  $F_p = 0.05$ . Figure 7.b shows the relation between the fraction of false positives ( $F_p$ ), and the memory (Mem) required by a bit filter and its selectivity (Sel). Thus, if SHJ was limited by the amount of memory, then a larger fraction of false positives would allow the necessary bit filters to fit in the local memory. Using less memory increases the number of false positives that can be done in a controlled manner. Sets 3 and 4 are always the worst case because *orders* and *ps\_partsupp*, which are the larger dimension tables, have high selectivities, and hence the size of the bit filters created during the respective build phases, is large because of the presence of a high of incoming distinct values.

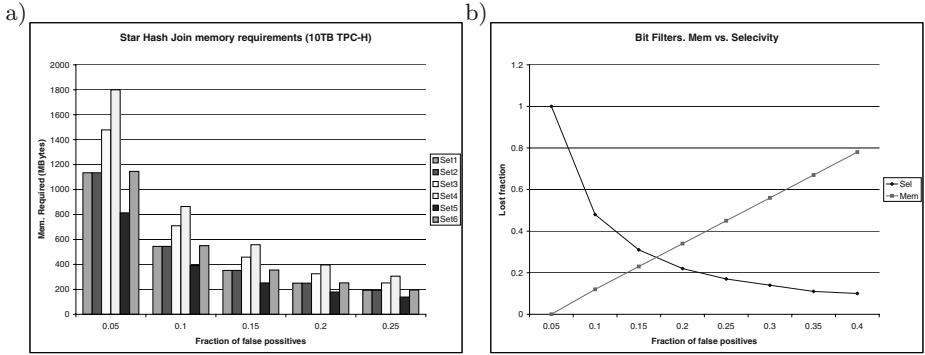


Fig. 7. Memory used by the Star Hash Join technique

## 5 Related Work

Ad hoc star join processing has been the focus of some research recently. Pre-computed aggregate results, the use of indices, and clustering schemes are the most relevant techniques to speed up query processing. The use of materialized views [18] in ad hoc star join queries is limited because it is not possible to know what is being queried in advance. BitMap Joins, introduced by O’Neil and Graefe in [16] are the basis for other research on the topic; these are based on the use of Bitmap indices [17][9]. The alternative to the use of indices is physical clustering which aims at limiting the number of I/O accesses to the Fact table required to process a query [12][14]. In this case, the Fact table may be created by specifying one or more key dimensions that are used to cluster the table data. Thus, the Fact table is organized in multiple hierarchical dimensions. Star joins are then turned into multidimensional range queries reducing the number of I/O accesses to the Fact table. A detailed comparison of these techniques and SHJ has been carried out in the extended version of this paper [4].

## 6 Conclusions

Star Hash Join (SHJ) generalizes Pushed Down Bit Filters (PDBF) to CLUMP architectures. Using the Semi-join reduction, bit filters are replicated in the memories of all clusters, and thus, PDBF can be used in such environments. Separately, they are totally different techniques: the former avoids data traffic communication by using the bit filters created in remote clusters; the latter, which can only be used in shared-everything systems, is aimed at avoiding data processing by pushing down the bit filters to the lower operations of the execution plan. SHJ exploits the use of bit filters, combining both techniques in CLUMP architectures.

The models of SHJ show a significant improvement over the models of the baseline and Semi-join reduction, in terms of data communication and I/O processing for ad hoc star join query processing. Remarkably, the results show that

there is a reduction of between 60% and 90% for typical queries compared to the baseline algorithm, which uses the bit filters for each individual join locally in each cluster. Moreover, SHJ is not expensive in terms of the memory usage, and it does not require the use of auxiliary static structures like indices and materialized views.

## References

1. www.top500.org. Top500 Supercomputer Sites.
2. www.tpc.org. Transaction processing and database benchmarks.
3. Josep Aguilar-Saborit, Victor Muntés-Mulero, and Josep-Lluís Llorca-Pey. Pushing down bit filters in the pipelined execution of large queries. *Euro-Par 2003.*, pages 328–337.
4. Josep Aguilar-Saborit, Victor Muntés-Mulero, Josep-Lluís Llorca-Pey, and Calisto Zuzarte. Ad-hoc star hash join processing in clusters of smp. *Technical Report. Universitat Politècnica de Catalunya UPC-DAC-RR-GEN-2005-4.*
5. Josep Aguilar-Saborit, Victor Muntés-Mulero, Josep-Lluís Llorca-Pey, Calisto Zuzarte, and Hebert Pereyra. On the use of bit filters in shared nothing partitioned systems.. *To appear in IWIA '05.*
6. P.A Bernstein and D.M.Chiu. Using semijoins to solve relational queries. *J.ACM*, 28(1):25–40, 1981.
7. P.A. Bernstein and N.Goodman. The power of natural joins. *SIAM J.Computi.*, 10:751–771, November 1981.
8. Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
9. C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *Proc. of the SIGMOD Conf. on the Management of Data*, pages 355–366, 1998.
10. S. Chaudhuri and U. Dayal. Data warehousing and olap for decision support (tutorial). In *SIGMOD Conference 1997: 507-508.*
11. S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. In *SIGMOD*, volume 26, pages 65–74, 1997.
12. P. Deshpande, K. Ramasamy, A. Shuckla, and J. F. Naughton. Caching multidimensional queries using chunks. *SIGMOD Conference*, pages 259–270, 1998.
13. David J. DeWitt, R. Katz, F. Olken, L. Shapiro, M. Stonebreaker, and D. Wood. Implementation Techniques for Main Memory Database Systems. In *Proceedings of the SIGMOD Int'l. Conf. on the Management of Data*, pages 1–8. ACM, 1984.
14. Volker Markl, Frank Ramsak, and Rudolf Bayer. Improving olap performance by multidimensional hierarchical clustering. *Proc. of the Intl. Database Engineering and Applications Symposium*, pages 165–177, 1999.
15. M. Mehta and David J. DeWitt. Parallel database systems : The future of high performance database processing. *Proceedings of the 21st VLDB Conference*, 1995.
16. P. O'Neil and G. Graefe. Multi-Table Joins Through Bitmapped Join Indices. *SIGMOD Record*, 24(3):8–11, 1995.
17. P. O'Neil and D. Quass. Improved query performance with variant indexes. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 38–49, 1997.
18. R. Roussopoulos. Materialized Views and Data Warehouses. *SIGMOD Record*, 27(1):21–26, 1998.

# A Precise Blocking Method for Record Linkage

Patrick Lehti and Peter Fankhauser

Fraunhofer IPSI, Dolivostr. 15, Darmstadt, Germany  
{Patrick.Lehti, Peter.Fankhauser}@ipsi.fraunhofer.de

**Abstract.** Identifying approximately duplicate records between databases requires the costly computation of distances between their attributes. Thus duplicate detection is usually performed in two phases, an efficient blocking phase that determines few potential candidate duplicates based on simple criteria, followed by a second phase performing an in-depth comparison of the candidate duplicates. This paper introduces and evaluates a precise and efficient approach for the blocking phase, which requires only standard indices, but performs as well as other approaches based on special purpose indices, and outperforms other approaches based on standard indices. The key idea of the approach is to use a comparison window with a size that depends dynamically on a maximum distance, rather than using a window with fixed size.

## 1 Introduction

The problem of identifying approximately duplicate records between databases is known, among others, as duplicate detection or record linkage. Traditional scenarios for duplicate detection are data warehouses, which are populated by several data sources. Analyses on the data warehouse influence business decisions, therefore a high data quality resulting from the data cleansing process is of high importance.

In general the duplicate detection process is performed in two phases consisting of a first phase identifying potential duplicates based on simple criteria, called blocking phase and a second phase performing an in-depth comparison of the potential duplicates resulting in the final estimation of a pair being a duplicate or not.

The main goal of the blocking phase is to reduce the number of detailed comparisons of records in the second phase, because even moderate sized data sets with  $10^5$  records would result in  $10^{10}$  record-pairs for a quadratic comparison of all possible pairs. Such a two phase approach was first used by Newcombe [1] and further formalized by the Fellegi-Sunter model for record linkage [2]. Most other recent work on duplicate detection also follows this approach, e.g. [3,4,5].

This paper presents a variant of the sorted-neighborhood method of Hernandez and Stolfo [3] for the blocking phase. This variant significantly decreases the number of false matches without increasing the number of false misses. This is accomplished by determining the size of the window based on a maximum distance (like edit-distance) dynamically, in contrast to the fixed sized window of the original approach.



The remainder of this paper is organized as follows. Section 2 lists requirements for the blocking phase and introduces related work. Section 3 presents our blocking approach and Section 4 discusses an efficient implementation of it. Section 5 presents the results of the evaluation of our approach compared to the original one and Section 6 concludes.

## 2 Blocking

The purpose of the blocking phase is to find all duplicate records with a very simple and efficient method. Because the second phase is in general very expensive computationally, the blocking phase should also not introduce too many false matches. Therefore the blocking phase should ideally fulfill the following requirements:

- no false misses (very high recall)
- few false matches (high precision)
- few and cheap comparisons (low cost)
- low or no user-interaction (easy configuration)

It can be easily seen that there is a trade-off between "high recall", "high precision" and "easy configuration". As the precision is increased in the detailed second phase, but the recall can not be increased later on, the priority should definitely be on the recall side. Fellegi-Sunter examine the effect on the level of false misses based on a blocking scheme [2]. A more detailed paper on the problem of how to select an appropriate blocking scheme is found in [6].

A standard blocking method is to use the classic key approach for record linkage [7], i.e. record equivalence is determined based on equivalence of key expressions. For blocking these keys are chosen to be very general in order to produce a high recall. Often such a key must be so unspecific that the resulting blocks are much too large. Additionally, small typos in the selected blocking key values result in records being put in different blocks and finding such a blocking key requires high user-interaction.

The sorted-neighborhood method [3] sorts the records based on a sorting key and then moves a window of fixed size sequentially over the sorted records. All pairs within such a window are classified as potential duplicates. The disadvantage of this approach is the fixed size of the window; if the number of records with the same sorting key is larger than the window size, not all potential duplicate records will be compared. Typos in the first characters of the sorting key can also result in records being outside the window and thus not be detected as potential duplicates. For this problem Hernandez-Stolfo [3] propose to do multiple passes with different sorting keys and small window sizes instead of one pass with a large window size.

This paper presents a modified version of the original sorted-neighborhood method by replacing the fixed sized window with a dynamically sized window based on a distance measure between the keys. The original sorted-neighborhood method is compared to our blocking approach in the evaluation.

The "Bigram Indexing" [8] method converts the blocking key values into a list of bigrams (sub-strings containing two characters) and builds sublists of all possible permutations using a threshold. The resulting bigram lists are sorted and inserted into an inverted index, which is used to retrieve the corresponding record numbers in a block.

Finally, the "Canopy Clustering" [9] method forms blocks of records based on those records placed in the same canopy cluster. Canopy clusters contain all records within a thresholded distance of a blocking key from a center record, which in general results in overlapping clusters. The distance is calculated using some distance measure like edit-distance.

Baxter et al. [10] compared these four blocking algorithms and showed that the "Bigram Indexing" and the "Canopy Clustering" significantly outperform the two more traditional approaches. We compare our approach to the results of their experiments.

### 3 Approach

We achieved the best results with a variant of the sorted-neighborhood method as presented by Hernandez-Stolfo [3]. In the case that a single data source (or already merged data sources) need to be examined for duplicates, the algorithm is shown in Figure 1.

```
function detectDuplicates(records S, key K)
  sort S on K

  init W with ()

  for each key1 in S
    for each key2 in W
      if(distance(key1, key2) < Threshold)
        then potential-duplicates(
          records of key1,
          records of key2
        )
      else remove key2 from W
  add key1 to W
```

Fig. 1. Blocking algorithm on single source

The records are sorted based on a key value, which can be a single attribute value or the result from a complex key expression. We try to use simple sorting keys, which in practice often allows to use existing indices on the key to get the sorted list of records. Otherwise the sorting is done by creating a new index for that key. The complexity to create such an index is in general  $O(n * \log n)$ .

The sorted list of records is then sequentially scanned for potential duplicates by comparing all keys within a sliding window. In contrast to Hernandez-Stolfo [3] we use a dynamically sized window, where the size depends on a fixed distance between the key values. The distance between the key values is determined by a distance function like edit-distance. This dynamically sized window is in practice often much smaller than a fixed sized window, but larger when required.

The complexity of this algorithm is  $O(n * \log n + k * w)$ , where  $k$  is the number of different key values and  $w$  is the average size of the window.  $k$  depends on the uniqueness of the selected blocking key. If two data sources or a new data source and a set of already integrated data sources need to be examined for duplicates, the algorithm can be slightly modified, that only keys from different data sources are compared, assuming the individual data sources are duplicate free. Also the sorting of the integrated data set needs to be done only once and is therefore ignorable in the long run, which further reduces the complexity of the algorithm.

As long as the sorting of the keys is in general not done by distance, but mostly alphabetically, it is clear that not all keys that are within the distance threshold are also within the window. In other words, for the three sorted keys with  $key1 < key2 < key3$  it does not necessarily hold that  $dist(key1, key2) < dist(key1, key3)$ . To this end, it is better to do multiple passes with different keys, instead of selecting just a single sorting key for the blocking process. Experiments showed that such a multi-pass approach provides higher recall and precision for the resulting potential duplicate set, even with much smaller window sizes and therefore fewer comparisons than a single-pass approach. This is consistent with the results of the original sorted-neighborhood method of Hernandez-Stolfo [3].

## 4 Evaluation

### 4.1 Datasets

For an evaluation we have chosen a *Restaurant*, a *Census* data set and a generated *Mailing* data set, which have been previously used as benchmarks for duplicate detection and blocking methods, e.g. in [5,4,10]. The restaurant dataset contains 864 restaurant names and addresses with 112 duplicates, composed of 533 and 331 restaurants assembled from Fodor’s and Zagat’s restaurant guides. These individual datasets are duplicate free. The attributes used in the experiments are restaurant name and street address. Table 1 shows a sample duplicate record from this dataset.

The census data set is a synthetic dataset containing 824 census-like records with 327 duplicates, composed of two duplicate free sets with 449 and 375 records. The attributes used in the experiments are last name and first name. Table 2 shows a sample duplicate record from this data set.

The *Mailing* data set is generated by the DBGen [3] database generator. It generates artificial address records and randomly introduces duplicates with errors. We have used the same settings as [10] to generate data sets with approximately 1000, 2000, 5000 and 10000 records, where the number of clusters

**Table 1.** Sample duplicate records from the *Restaurant* data set

name	address	city	cuisine
uncle nick's	747 ninth ave.	new york city	greek
uncle nick's	747 9th ave. between 50th and 51st sts.	new york	mediterranean

**Table 2.** Sample duplicate records from the *Census* data set

last name	first name	house number	street
JIMENCZ	WILLPAMINA	S 214	BANK
JIMENEZ	WILHEMENIA	214	BANKS

**Table 3.** Sample duplicate records from the *Mailing* data set

last name	first name	street number	street	city
Swenberg	Gruemnkranz	436	Klich Avenue	Anchor Point
Swenbearg	Gruenkranz	436	Klich Avenue	sAnchor Point

are half the number of records, which results in most records having a single duplicate. However some records will have more than one other record with a true match and some will have no duplicates. For the experiments only the last name and first name is used. Table 3 shows a sample duplicate record from this dataset.

## 4.2 Experimental Methodology

The data sets are transformed into an XML format and stored into an XML database. Indices on every single attribute were created, which corresponds to a sorting on every attribute.

For comparison of the experimental results with [10], we use the "reduction ratio" (RR), "pairs completeness" (PC) and "F score" as defined there, instead of the usual precision, recall and F-measures as used in information retrieval [11]

$$RR = 1 - \frac{|PotentialDuplicates|}{|AllPairs|}$$

$$PC = \frac{|CorrectlyIdentifiedDuplicates|}{|TrueDuplicates|}$$

$$Fscore = \frac{2 * RR * PC}{RR + PC}$$

The reduction ratio is the relative reduction in the number of pairs to be compared. The pairs completeness is equivalent to the definition of recall. RR and PC should be maximized, however there is a tradeoff between them. The F score is a single measure that combines RR and PC via a harmonic mean. Instead of precision-recall curves we present RR-PC curves for a corresponding graphical overview.

### 4.3 Comparison with the Original Method

At first we compare our method to the original sorted-neighborhood method [3]. To this end we measure the RR and PC values for the original fixed window

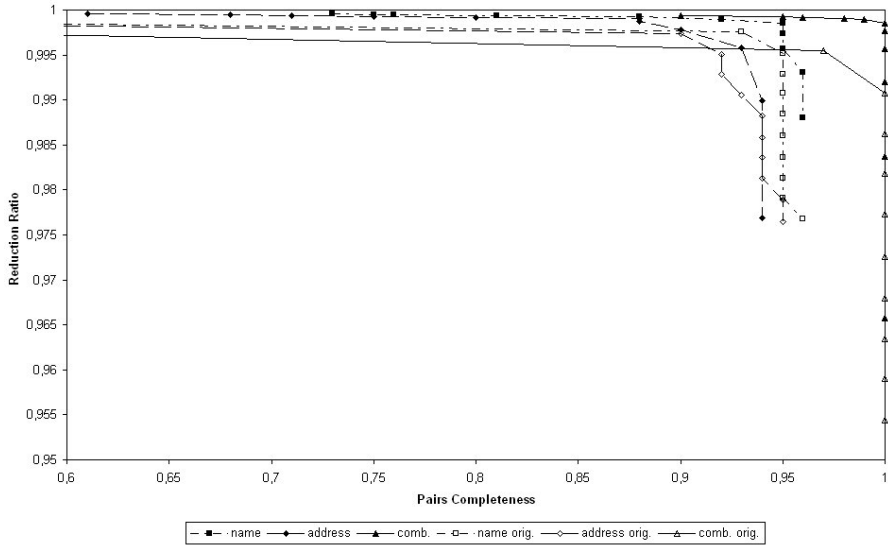


Fig. 2. RR-PC for the restaurant data set

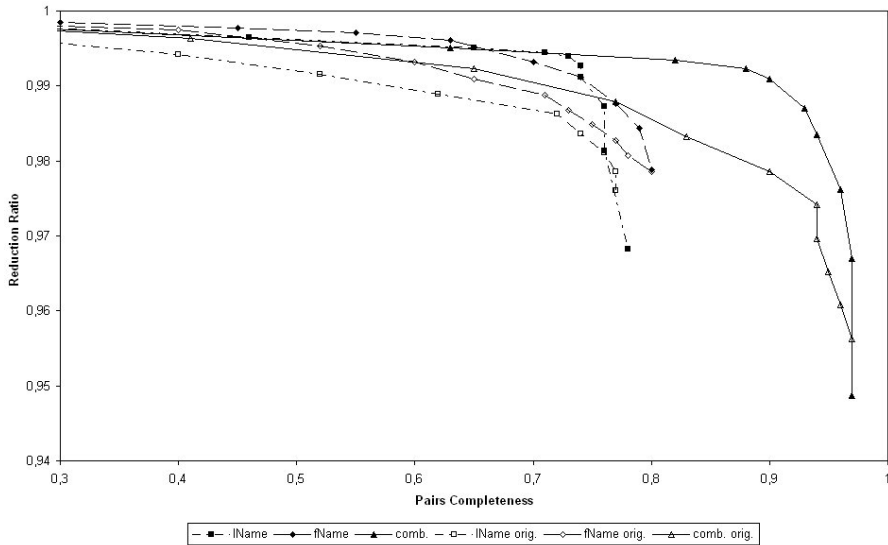


Fig. 3. RR-PC for the census data set

sizes between 0 and 10 in single steps and for our dynamically sized windows for distances between 0 and 0.5 in steps of size 0.05. Both methods are used in a single pass way on the two individual attributes as well as in a two pass way on both attributes. The RR-PC curve for the restaurant data set can be seen in Figure 2, the RR-PC curve for the census data set in Figure 3.

Both figures show that our method significantly outperforms the original method in all experiments. It can be further seen that the combined multi-pass approach reaches a much higher accuracy than the single-pass approaches. E.g. the combined approaches on the restaurant data set show a reduction ratio of 0.999 for our vs. 0.991 for the original approach for the maximum pairs completeness; on the census data set our approach reaches 0.967 vs. 0.956 of the original approach.

#### 4.4 Costs

In order to compare the costs of the original method to our approach, we measure the number of distance calculations between individual attributes. Our approach needs to calculate such distances during blocking for determining the window, but at the same time it is able to save this distance in the comparison vector of the potential duplicates for the second duplicate detection phase. In contrast, the original approach needs no such comparisons during the blocking phase, but has to determine the complete comparison vector distances in the second phase. Therefore a valid cost comparison is to count the number of distance calculations during the blocking and the second phase. Figure 4 shows the number of calculations in comparison to the reached pairs completeness for the restaurant data set and Figure 5 shows the same for the census data set.

These figures show that our approach is either equally expensive or even cheaper in terms of number of comparisons. E.g. the combined approaches on the restaurant data set show 2048 comparisons for our vs. 3244 comparisons for the original approach for the maximum pairs completeness; on the census data set our approach needs 10365 vs. 14732 comparisons for the original approach for the maximum pairs completeness. This reveals that moving a part of the comparison complexity already to the blocking phase does not only significantly increase the accuracy of the resulting potential duplicates, but also reduces the overall costs of the duplicate detection process.

#### 4.5 Comparison with Other Blocking Methods

Baxter et al. [10] have compared the standard blocking using keys, the original sorted-neighborhood, the Bigram indexing and the Canopy Clustering methods against the mailing data set. In order to compare our approach with their results, we have conducted experiments with the same settings for three different dynamic window sizes: 0.1, 0.2 and 0.3.

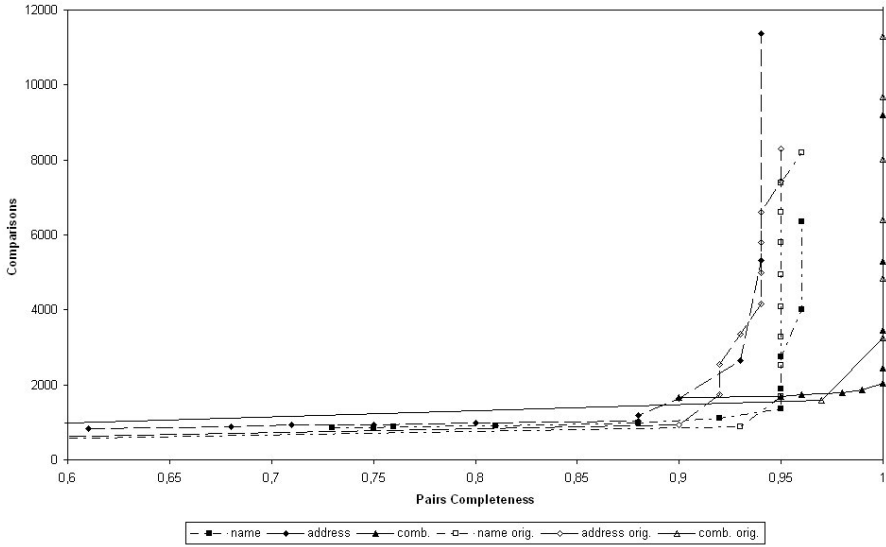


Fig. 4. Blocking costs for the restaurant data set

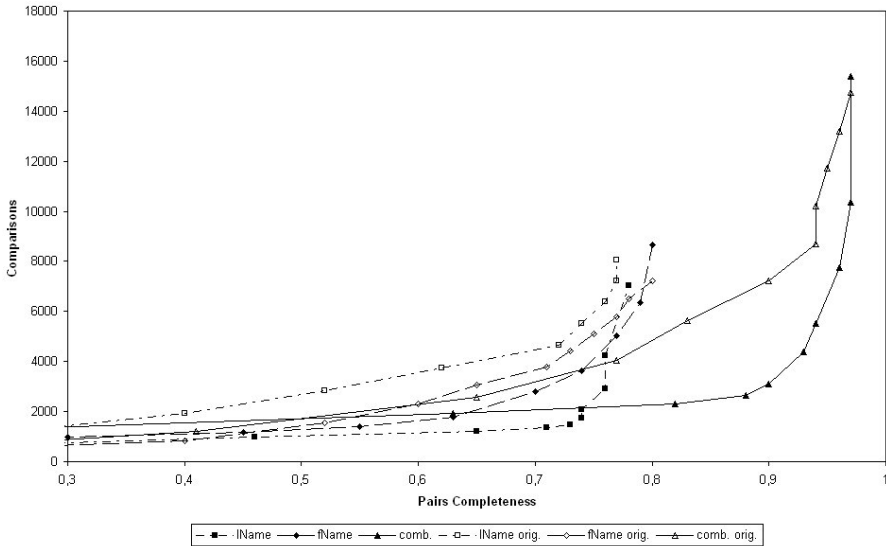


Fig. 5. Blocking costs for the census data set

Figure 6 shows the pairs completeness, Figure 7 shows the reduction ratio and Figure 8 the F score against the size of the data sets for the three window sizes and the two best other approaches from [10], that are the Canopy Clustering with cluster size 1.5 and Bigram Indexing with bigram size of 0.3.

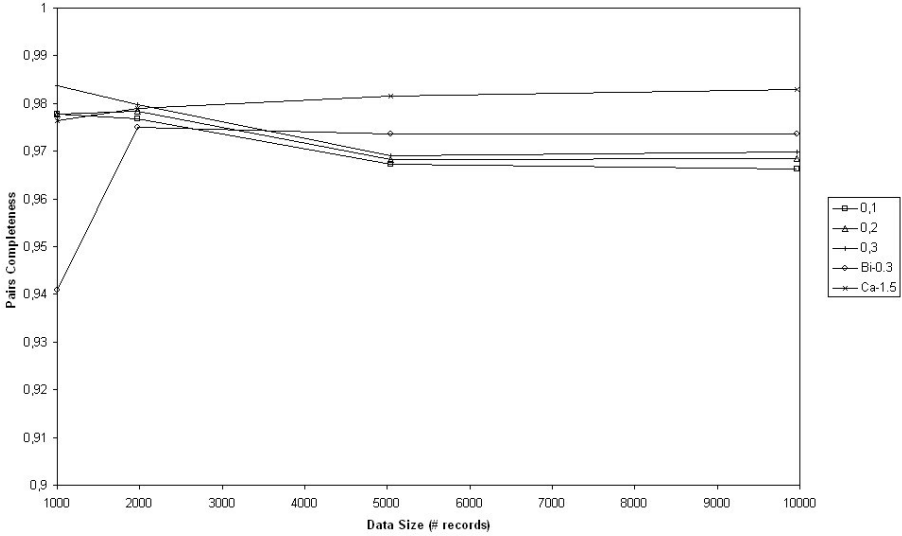


Fig. 6. Pairs completeness on the mailing data set

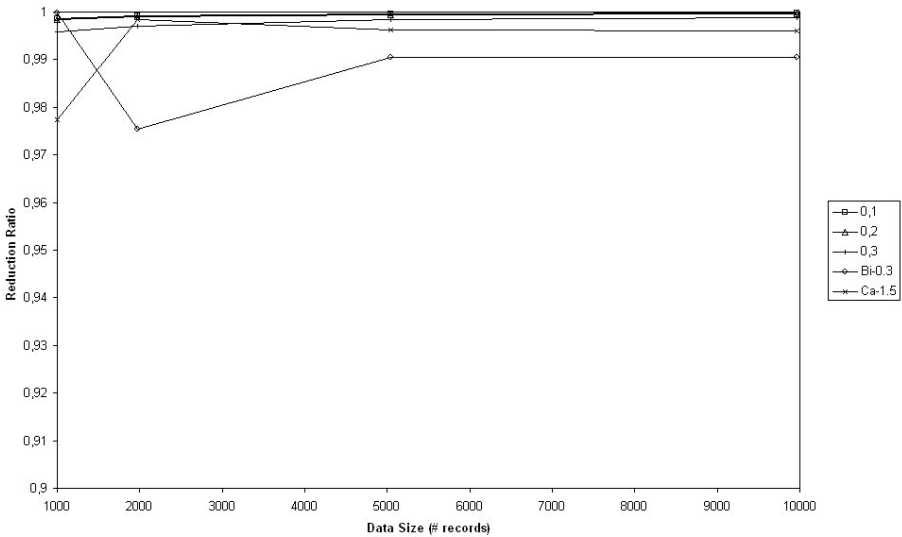
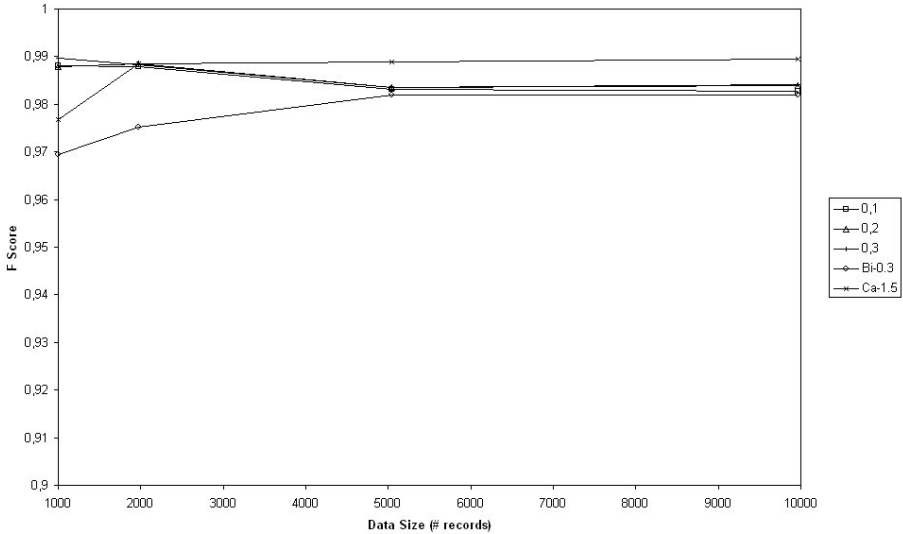


Fig. 7. Reduction ratio on the mailing data set

These figures clearly show that our approach is competitive for all window sizes with even the best other approaches. In contrast to the other top blocking methods, our approach does not require any new indexing structures to be implemented. It simply uses existing standard indices on individual attributes.





**Fig. 8.** F score on the mailing data set

## 5 Conclusion

This paper has presented a variant of the sorted-neighborhood method for the blocking phase in record linkage. Our approach removes the main disadvantage of the method by replacing the fixed sized windows with dynamically sized windows using distance measures. The evaluations have shown that our approach significantly improves the accuracy of the original method and does this surprisingly without higher costs. Further experiments have shown that our approach is also competitive to even the best other blocking methods, without the need for any additional indexing structures, but simply relying on standard indices on individual attributes. As future work we want to automatically find an optimal threshold for the dynamic window size for a given key.

## Acknowledgement

This research is supported by the bmb+f in the SemIPort project.

## References

1. Newcombe, H.B., Kennedy, J.M., Axford, S.J., James, A.P.: Automatic linkage of vital records. *Science* **130** (1959) 954–959
2. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. *Journal of the American Statistical Association* **64** (1969) 1183–1210

3. Hernandez, M.A., Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery* **2** (1998) 9–37
4. Ravikumar, P., Cohen, W.W.: A hierarchical graphical model for record linkage. In: *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, AUAI Press (2004) 454–461
5. Bilenko, M., Mooney, R.J.: Learning to combine trained distance metrics for duplicate detection in databases. Technical Report AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX (2002)
6. Kelley, R.P.: Advances in record linkage methodology: a method for determining the best blocking strategy. In: *Proc. of the Workshop on Exact Matching Methodologies in Arlington - Record Linkage Techniques*. (1985)
7. M.Jaro: Advances in record linkage methodology as applied to matching the 1985 census of tampa. *Journal of the American Statistical Society* **84** (1989) 414–420
8. Christen, P., Churches, T.: *Febrl: Freely extensible biomedical record linkage manual*, release 0.2 edition. <https://sourceforge.net/projects/febrl/> (2003)
9. McCallum, A., Nigam, K., Ungar, L.: Efficient clustering of high-dimensional data sets with application to reference matching. In: *Proc. of the sixth ACM SIGKDD Int. Conf. on KDD*. (2000)
10. Baxter, R., Christen, P., Churches, T.: A comparison of fast blocking methods for record linkage. In: *Proceedings of the Workshop on Data Cleaning, Record Linkage and Object Consolidation at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (2003)
11. Baeza-Yates, R., Ribiero-Neto, B. In: *Modern Information Retrieval*. Addison Wesley (1999) 74–79

# Flexible Query Answering in Data Cubes

Sami Naouali and Rokia Missaoui

LARIM, Department of Computer Science and Engineering,  
University of Quebec in Outaouais, C.P. 1250, Succ. B,  
Gatineau (Qc), Canada, J8X 3X7  
{Sami.Naouali, Rokia.Missaoui}@uqo.ca

**Abstract.** This paper presents a new approach toward approximate query answering in data warehouses. The approach is based on an adaptation of rough set theory to multidimensional data, and offers cube exploration and mining facilities.

Since data in a data warehouse come from multiple heterogeneous sources with various degrees of reliability and data formats, users tend to be more tolerant in a data warehouse environment and prone to accept some information loss and discrepancy between actual data and manipulated ones.

The objective of this work is to integrate approximation mechanisms and associated operators into data cubes in order to produce views that can then be explored using OLAP or data mining techniques. The integration of data approximation capabilities with OLAP techniques offers additional facilities for cube exploration and analysis.

The proposed approach allows the user to work either in a *restricted* mode using a cube *lower approximation* or in a *relaxed* mode using cube *upper approximation*. The former mode is useful when the query output is large, and hence allows the user to focus on a reduced set of fully matching tuples. The latter is useful when a query returns an empty or small answer set, and hence helps relax the query conditions so that a superset of the answer is returned. In addition, the proposed approach generates classification and characteristic rules for prediction, classification and association purposes.

**Keywords:** approximate query answering, OLAP queries, data mining, rough set theory, data warehouses, multidimensional data.

## 1 Introduction

Since data in data warehouses (DW) come from multiple heterogeneous sources with different levels of reliability and quality, users may be more tolerant and accept some information loss and discrepancy between actual data and manipulated ones provided a more efficient query processing is insured. In relational database literature, there have been a set of studies about flexible query answering (see for e.g., [1,2]). They mainly focus on relaxing query conditions in order to return non-empty answer sets.

In the context of DWs and multidimensional data, query approximation has been used in order to accelerate aggregate computation and query execution at the expense of some information loss. Most of work has been conducted based on sampling techniques [3,4]. In [5], a wavelet-based approach is used for approximate query answering and proves to be more effective than sampling techniques. In a similar spirit, [6] uses the probability density distribution of data in order to propose a compressed representation of data cubes which reduces data storage and leads to approximate answers of aggregate queries. Wavelet based techniques have been used either for progressive evaluation of some specific OLAP queries [7,5] or as a sampling technique. As in [6], Vitter *et al.* [8] uses wavelets for compressing sparse data cubes and getting approximate answers of aggregate queries.

In our approach, we allow an approximate evaluation of OLAP queries such that the output is either a subset or a superset of the exact query answer.

Our present work aims at offering flexible query answering mechanisms and tools for cube exploration. To that end, we have (i) adapted the rough set theory to multidimensional data in order to provide approximate answers to queries and define concepts (cube subsets) according to user's need (ii) proposed an enrichment of OLAP techniques with new operators that allow more flexible interactions between the user and the data warehouse, and (iii) defined materialized views to capture and then exploit the output of approximate operators for query answering and data mining purposes (cell clustering and association rule mining). As a consequence, the approximation concerns not only the answer to OLAP queries but also the data mining output.

The paper is organized as follows. Section 2 provides a brief background about the rough set theory while Section 3 presents an illustrative example. The proposed approach is described in Section 4.

## 2 Background

The Rough Set Theory (RST) was introduced by Z. Pawlak at the beginning of the 80s [9]. It offers a theoretical foundation for handling vague concepts and badly defined borders [10]. Its emergence represents an important evolution in the area of artificial intelligence and notably in inductive learning from incoherent data. It was applied in a wide variety of applications such as medicine, industry, finance, commerce, and so on.

The theory of rough sets is based on the notions of indiscernibility and approximation. The former expresses the degree of similitude between objects while the latter allows a description of a concept (a given set of objects), based on possible flaw in data.

In an approximation context, an information system  $\mathcal{A}$  is a pair  $(U, A)$  where  $U$  is a non-empty finite set of objects called the universe and  $A$  is a non-empty finite set of attributes.

Function  $f$  returns the value of an attribute in  $A$  for any object in  $U$ :

$$f : U \times A \rightarrow V$$

$$\forall o \in U \text{ and } a \in A, (o, a) \mapsto f(o, a) \in V_a$$

where  $V_a$  is the value set of  $a \in A$  and  $V$  is the union of the value sets attached to the attributes in  $A$ .

Indiscernibility is an equivalence relation upon a subset  $P$  of attributes used to describe the objects in the universe. Two objects are *indiscernible* with respect to  $P$  if they have the same value for each attribute in  $P$ . In order to conduct approximation and data mining tasks, two disjoint subsets are extracted from the whole set  $A$ : the set  $C$  of *condition* (description) attributes used to describe the objects, and the set  $D$  of *decision* (classification) attributes used to partition the universe into several concepts (a set of objects) in which objects will be approximately classified.

Given two objects  $o_i$  and  $o_j \in U$ , the indiscernibility relation  $I_P \subset U \times U$  w.r.t.  $P \subseteq C$  is defined as:

$$I_P = \{(o_i, o_j) \in U^2 : \forall q \in P f(o_i, q) = f(o_j, q)\} \quad (1)$$

where  $f(o, q)$  is the value of  $q$  for the object  $o$ . Since the indiscernibility relation is an equivalence one, the universe  $U$  can be partitioned into equivalent classes in which objects are indiscernible from each other by attributes from  $P$ .

For any given concept  $X$ , the RST allows the computation of two approximate sets that are a subset and a superset of the exact set of objects respectively (see Figure 2 for an illustration). The first one is called the *lower approximation* and contains objects that belong actually to the concept  $X$ , whereas the second one is called the *upper approximation* which may contain objects that are not elements of the described concept but are indiscernible with one or several objects associated with  $X$ . From these two approximations, one can compute additional approximate sets such as the boundary region as well as the positive and negative regions. It is also possible to compute classification and characteristic rules. More details will be given in Section 4.

A generalization of the RST is called the  $\alpha$ -Rough Set Theory ( $\alpha$ -RST) [11,12] where  $\alpha$  represents the tolerated degree of similarity between objects with respect to a given subset of description attributes (see Section 4 for more details).

### 3 An Illustrative Example

The following example will serve as an illustration for the proposed approach. It concerns a simplified structure of a data warehouse, called *Patents-OLAP*, related to patent submission and approval. For simplicity reasons, we limit ourselves to a data cube relating a fact table to the following dimension tables (see Figure 1) and having the number of patents (*nb\_patents*) as the unique measure:

- APDM (APplication Date in Month): month when the patent is applied for.
- APDY (APplication Date in Year): year when the patent is applied for.
- ISDM (ISsue Date in Month): month when the decision (rejection or approval) about the patent is issued.
- ISDY (ISsue Date in Year): year when the decision is given.
- ET (Elapsed Time): elapsed time in month intervals between submission and decision dates.

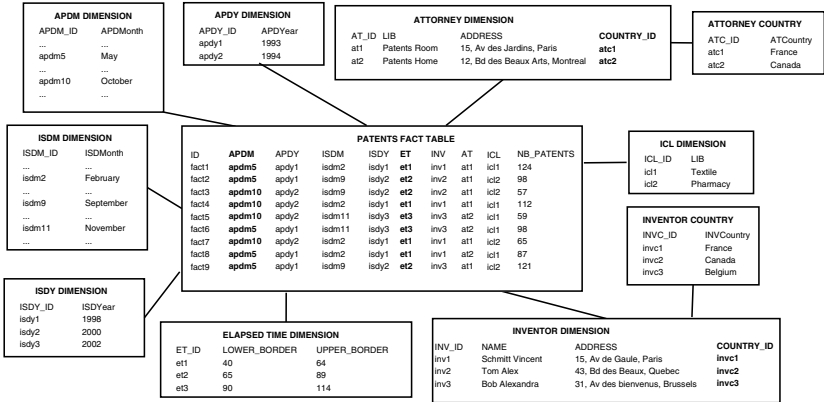


Fig. 1. A snowflake schema and extent of Patents-OLAP data cube

- INV (INventor): information about the patent inventor.
- AT (ATtorney or agent): information about the legal representative of the inventor.
- ICL (International CLassification): international nomenclature of the patent.

Since the classification of patents is manually conducted by domain experts, some patents could actually belong to a classification that is different from the one selected by humans. To illustrate this fact, let us assume that an inventor is applying for two patents, one for a given drug and another one for packing it in an appropriate way. We also assume that the two patent applications are handled by the same legal representative and have very similar properties (inventor, legal representative, application date and probably decision date and elapsed time). According to the selected variables, the two patent applications (facts) may be indiscernible and hence belong to the same class while experts classified the first patent in the *pharmacy* class, and the second one in the *paper and printing* class. The discovered indiscernibility may give a *hint* for a classification revision of patent applications. Therefore, our approach can be exploited to revise the classification proposed by experts by highlighting the similarity that may hold between facts for a given set of dimensions corresponding to a given user’s perspective.

## 4 An Approximation Method

Our contribution consists to first adapt the rough set basics to the framework of multidimensional data. Then, mechanisms for approximate query answering as well as OLAP analysis and data mining are settled.

#### 4.1 Adaptation of the $\alpha$ -RST to the Multidimensional Data Framework

$\alpha$ -RST offers more flexibility than RST in the sense that indiscernibility between two objects is assessed according to a subset (instead of the whole set) of condition attributes.

In this section we present the integration of the rough set theory with the multidimensional framework by proposing new operators to compute concept approximations (see Section 4.3). This can be perceived as an enrichment of OLAP techniques with uncertainty handling mechanisms. Such enrichment introduces flexibility in querying multidimensional data since noisy/dirty data (e.g., missing values) can be taken into account.

The multidimensional model of a data warehouse contains mainly a fact table  $FT$  linked to a set of dimensions  $DIM = \{dim_1, \dots, dim_n\}$ . To each  $dim_i$  corresponds a set of values, i. e., *members*, that we note  $dim_{i_1}, \dots, dim_{i_m}$ . In such multidimensional context, the universe mentioned in the rough set theory corresponds to the fact table  $FT$  and hence an object is a fact, and key-attributes are obviously logical pointers to dimension tables. Dimensions are divided into two distinct sets: *condition dimensions*  $C^1$  (used to describe facts) and *decision dimensions*  $D^2$  that will be used for classification and prediction purposes. To a given dimension from  $D$  may correspond as many *target concepts* (i.e., sets of facts that belong to a given class) as there are members (e.g., ICL nomenclature = textile) of this decision (classification) dimension. Hence, computing approximations of this dimension consists in describing the appurtenance of facts to each one of the target concepts associated with  $D$ .

*Example 1.* Using the illustrative example, one can select ICL as the decision dimension and the rest of dimensions as condition (description) dimensions. By doing so, we aim at describing the outbuilding of patents to a given international class not only based on the nature of the patented product, but also on information about the application date, the time needed to issue a patent decision, the inventor as well as the legal representative of the inventor. Then, we can conduct a kind of superposition of patent classification conducted by domain experts and patent classification provided by our approach.

By relying on approximation mechanisms offered by the RST one can generate two classes in which a given patent may be put: a "strict" class in which a subset of a given target concept  $X$  can be placed with certainty, and a "relaxed" class containing all the elements of  $X$  plus patents having indiscernible members outside that concept. The user can decide, according to his needs and the degree of flexibility he initially sets up, to consider one approximation or another to describe the sought concept, rather than considering the set  $X$  of patents that a conventional query system would return.

---

<sup>1</sup>  $C = \{c_1, c_2, \dots\}$  with  $dom(c_i) = \{c_{i_1}, c_{i_2}, \dots\}$ .

<sup>2</sup>  $D = \{d_1, d_2, \dots\}$  with  $dom(d_i) = \{d_{i_1}, d_{i_2}, \dots\}$ .

Figure 1 shows that the decision dimension ICL has two distinct values:  $icl_1$  (*textile*) and  $icl_2$  (*pharmacy*). Then, two target concepts are produced, *i.e.*,  $ICL = textile$  and  $ICL = pharmacy$ .

Let us now consider two facts  $x, y \in FT$ ,  $I_P^\alpha$  an indiscernibility relation built on  $FT$  according to a dimension set  $P \subseteq C$  and defined as follows:

$$xI_P^\alpha y \Leftrightarrow \frac{|\{q \in P : f(x, q) = f(y, q)\}|}{|P|} \geq \alpha \tag{2}$$

where  $\alpha \in [0, 1]$  defines the minimal proportion of condition dimensions that the facts  $x$  and  $y$  have to share to be considered as indiscernible<sup>3</sup>. As opposed to the classical rough set theory in which indiscernibility relation w.r.t. a concept  $P$  generates a partition of equivalence classes,  $\alpha$ -RST leads to an overlapping of classes.

*Example 2.* Let  $\alpha = 0.75$ , and  $P = C = \{APDM, ET, INVC, ATC\}$ . The two facts  $fact_2$  and  $fact_9$  belong to the same indiscernibility relation since they share the same values for 0.75% of the four dimensions.

The lower approximation of  $X \subseteq FT$  according to  $P$ , noted  $\underline{P}(X)$ , is the union of all the equivalence classes  $G_k$  defined according to  $I_P^\alpha$  and included in  $X$ :

$$\underline{P}(X) = \bigcup_{G_k \subseteq X} G_k \tag{3}$$

In other words, the lower approximation of a fact subset can contain only facts whose appurtenance to the target concept is certain, *i.e.*, for any  $x_i \in \underline{P}(X)$  one can assert that  $x_i \in X$ .

The upper approximation of  $X$  according to  $P$ , noted  $\overline{P}(X)$ , is the union of the equivalence classes defined according to  $I_P^\alpha$  and containing at least one fact of  $X$ :

$$\overline{P}(X) = \bigcup_{G_k \cap X \neq \emptyset} G_k \tag{4}$$

The upper approximation of  $X$  can then contain facts which are not necessarily included in  $X$  but are indiscernible with one (or several) facts of  $X$ . Therefore, lower and upper approximations of a set  $X$  are respectively the interior and the closure of this set in the topology generated by the indiscernibility relation as illustrated by Figure 2.

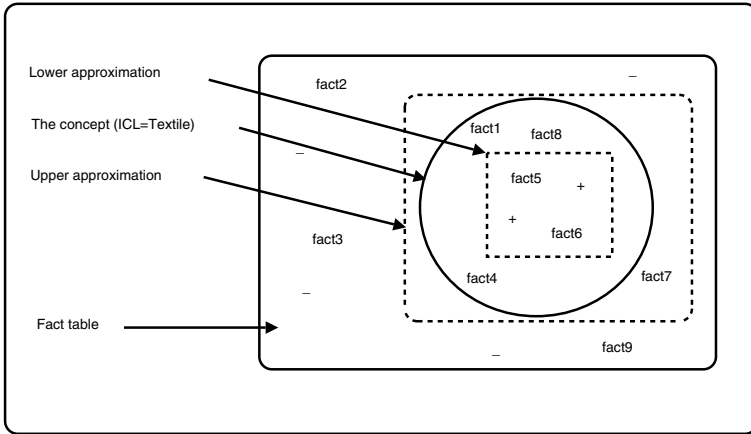
The boundary region of  $X \subseteq FT$  based on  $P$ , denoted by  $\tilde{P}(X)$ , contains objects that cannot be put into  $X$  in a conclusive manner. It is defined as follows:

$$\tilde{P}(X) = \overline{P}(X) - \underline{P}(X) \tag{5}$$

When the boundary region of  $X$  is non-empty (respectively empty), the set  $X$  is called *rough* (respectively *crisp*).

<sup>3</sup> when  $\alpha = 1$ ,  $I_P^\alpha$  is nothing but  $I_P$  described in Section 2.





**Fig. 2.** Approximate spaces corresponding to the concept  $ICL=Textile$

*Example 3.* Using the illustrative example, the set of equivalence classes extracted from the fact table with respect to the condition dimension set  $P = \{APDM, ET, ATC, INVC\}$  and  $\alpha = 0.75$  is:

$$\{\{fact_1, fact_4, fact_7, fact_8\}, \{fact_2, fact_3, fact_9\}, \{fact_5, fact_6\}\}.$$

For the target concept  $X$  corresponding to  $ICL = textile$ , the associated fact set to be approximated is  $X = \{fact_1, fact_4, fact_5, fact_6, fact_8\}$ . The lower approximation of  $X$  is the union of equivalence classes that are completely included in that concept, *i.e.*,  $\underline{P}(ICL = textile) = \{fact_5, fact_6\}$ . Its upper approximation is the union of equivalence classes which have a non-empty intersection with that target concept, *i.e.*  $\overline{P}(ICL = textile) = \{fact_1, fact_4, fact_5, fact_6, fact_7, fact_8\}$ . The boundary region of this concept is  $\tilde{P}(ICL = textile) = \{fact_1, fact_4, fact_7, fact_8\}$ .

Figure 2 illustrates this concept approximation by showing how the patents are spread over the approximation spaces corresponding to the given concept. The sign "+" represents patents that belong with certainty to the target concept, the sign "-" represents patents that are not members of that concept while the difference between the two sets delimited by dotted lines represents the boundary region, *i.e.*, patents that may belong to the concept but not with certainty.

In a similar way, the target concept  $ICL = pharmacy$  whose corresponding facts are in the set:  $\{fact_2, fact_3, fact_7, fact_9\}$  has a lower approximation  $\underline{P}(ICL = pharmacy) = \{fact_2, fact_3, fact_9\}$  and an upper approximation  $\overline{P}(ICL = pharmacy) = \{fact_1, fact_2, fact_3, fact_4, fact_7, fact_8, fact_9\}$ . Its boundary region is given by  $\tilde{P}(ICL = pharmacy) = \{fact_1, fact_4, fact_7, fact_8\}$ .

Let  $d_{i_1}, \dots$ , and  $d_{i_m}$  be values (members) of the target dimension  $d_i \in D$ , the positive region of  $d_i$  according to  $P$ , noted  $POS_P(d_i)$ , contains facts that can be put with certainty and no ambiguity in one of the target concepts that it generates. It is defined by:

$$POS_P(d_i) = \bigcup_{j:1,\dots,m} \underline{P}(d_{i_j}) \tag{6}$$

*Example 4.* The positive region for the decision dimension *ICL* is  $POS_P(ICL) = \{fact_2, fact_3, fact_5, fact_6, fact_9\}$ .

The negative region of  $d_i$  according to  $P$ , noted  $NEG_P(d_i)$ , contains facts that do not belong to any target concept associated with  $d_i$  and is defined by:

$$NEG_P(d_i) = FT - \bigcup_{j:1,\dots,m} \overline{P}(d_{i_j}) \tag{7}$$

Once the lower and upper approximations of each target concept are defined, association rules can then be generated. The lower approximation allows the generation of *classification rules*, called also *decision rules*, which are defined from the description of facts whose appurtenance to the target concept does not create any ambiguity. For a given decision dimension  $d_i$  and condition dimensions in  $P$ , the whole set of classification rules is computed from the positive region  $POS_P(d_i)$ . Rules generated from the upper approximation of a target concept are called *characteristic rules* since they are based on the description of facts whose appurtenance to the target concept is possible but not necessarily certain.

A classification rule is an expression of the form  $\varphi \implies (d_i = d_{i_j})$  drawn from the lower approximation, where  $d_i$  is a decision dimension,  $d_{i_j}$  a given value (member) of that dimension, and  $\varphi$  is a disjunction of predicates defining description attributes. Each predicate in  $\varphi$  can be a conjunction of conditions  $c_k = c_{k_l}$  where  $c_k \in C$  (condition dimensions) and  $c_{k_l}$  is a value of  $c_k$ . A characteristic rule is expressed by  $(d_i = d_{i_j}) \implies \varphi$  drawn from the upper approximation.

*Example 5.* In our example, the classification rules are:

$$\begin{aligned}
 & (APDM = October \vee May) \wedge (ET = 90 \text{ to } 114 \text{ months}) \wedge (INVC = Belgium) \\
 & \wedge (ATC = Canada) \longrightarrow (ICL = Textile)
 \end{aligned}$$

$$\begin{aligned}
 & ((APDM = May \vee October) \wedge (ET = 65 \text{ to } 89 \text{ months}) \wedge (INVC = Canada) \\
 & \wedge (ATC = France)) \vee ((APDM = May) \wedge (ET = 65 \text{ to } 89 \text{ months}) \\
 & \wedge (INVC = Belgium) \wedge (ATC = France)) \longrightarrow (ICL = Pharmacy)
 \end{aligned}$$

The first rule can be stated as follows: IF the patent application was submitted on October or May AND the decision was issued from 90 to 114 months AND the inventor’s country is Belgium AND the country of the legal representative is Canada, THEN its international classification is *textile*. Such rule has been generated from the lower approximation  $\underline{P}(ICL = textile) = \{fact_5, fact_6\}$ .

As a conclusion, the proposed approach allows the user to get an approximate answer to his query either in a *restricted* mode using a cube *lower approximation* or in a *relaxed* mode using cube *upper approximation*. The former mode is useful when the query output is large, and hence allows the user to focus on a reduced set of fully matching cells. The latter is useful when a query returns an empty or small answer set and allows to relax the query conditions. The exact answer lies between the lower and upper approximations [13].

## 4.2 Algorithm

The following algorithm for concept approximation in multidimensional data computes in "one shot" all the approximation spaces (upper, lower and boundary region) for each target concept associated with the decision dimension. It also computes positive and negative regions of that dimension and generates the corresponding classification and characteristic rules.

The algorithm makes use of the dimension tables as well as the fact table, takes into consideration the set  $P$  of description dimensions, the decision dimension  $d$  as well as a value for  $\alpha \in [0, 1]$ .

---

### Algorithm 1 $\alpha$ -approximations

---

```

1: input
2:  $FT$ : fact table with  $n$  dimensions and  $k$  measures;
3:  $DIM$ : set of  $n$  dimension tables ( $DIM = C \cup \{d\}$ );
4:  $P$ : description dimension subset,  $P \subseteq C \subset DIM$ ;
5:  $d$ : decision dimension with  $m$  members, i.e.,  $dom(d) = \{d_1, \dots, d_m\}$ ;
6:  $\alpha$ : similarity coefficient;
7: begin{ $\alpha$ -approximations}
8: for  $i$  from 1 to  $m$  do  $D_i = compute\_target\_concept(FT, d);$  /*  $D_i$  target concept
   of  $d^*$  /
9: for all  $D_i \in D_1, \dots, D_m$  do
10:  $D_i^\oplus := compute\_examples(D_i, FT);$  /*  $D_i^\oplus$  fact set (called examples) with the
   value  $d_i$  for  $d^*$  /
11: for all  $fact \in D_i^\oplus$  do
12:    $[fact]_{I_P^\alpha} := compute\_equivalence\_class(fact, FT, P, \alpha);$  /*  $[fact]_{I_P^\alpha}$  is the
   equivalence class of  $fact$  w.r.t.  $P^*$  /
13:   if  $[fact]_{I_P^\alpha} \subseteq D_i^\oplus$  then
14:      $insert([fact]_{I_P^\alpha}, LowerView\_D_i);$ 
15:   else
16:      $insert([fact]_{I_P^\alpha}, UpperView\_D_i);$ 
17:   end if
18: end for
19:  $BoundaryRegionView\_D_i := UpperView\_D_i - LowerView\_D_i;$ 
20: end for
21:  $ViewPOS_{P\_d} := \bigcup_{i=1}^n LowerView\_D_i;$ 
22:  $ViewNEG_{P\_d} := FT - \bigcup_{i=1}^n UpperView\_D_i;$ 
23:  $Table\_ClassificationRules\_d := generate\_classification\_rules(ViewPOS_{P\_d});$ 
24:  $Table\_CharacteristicRules\_d := generate\_characteristic\_rules(\bigcup_{i=1}^n UpperView\_D_i);$ 
25: end{ $\alpha$ -approximations}

```

---

The algorithm has two main steps. At the first step (lines 8 to 20), it computes target concepts associated with the decision dimension. A target concept (e.g., all patents that have  $ICL = Textile$ ) is noted  $D_i$  which corresponds to  $d = d_i$  where  $d$  is the target dimension and  $d_i$  the  $i^{th}$  value of the member set of  $d$ .

Then, for each target concept  $D_i$ , the algorithm computes the set of all facts having  $d = d_i$  (line 10). Then, the equivalence classes are computed<sup>4</sup> in line 12. If elements of an equivalence class are included in the current concept, then that class is added to the lower approximation of that concept. Otherwise, it is added to its upper approximation. At the end of this first step, and before exploring the next target concept, the algorithm computes (line 19) the view related to the boundary region associated with the current concept.

Once the target concepts are processed, positive and negative regions for the decision dimension as well as classification and characteristic rules can be computed using the views created at the first step.

The algorithm computes approximation spaces which are then stored as materialized views sharing the same structure as the initial fact table. Such views become an important component of the data warehouse, and are automatically linked to dimension tables. Moreover, they allow the user to create new OLAP cubes that encapsulate lower, upper and boundary spaces of target concepts as well as positive and negative regions of the target dimension.

The advantages of the proposed algorithm concern (i) the computation of approximate spaces related to a given decision dimension with respect to a set of characteristic dimensions, (ii) the relaxation of the similarity among facts through the parameter  $\alpha$ , (iii) the data mining of classification and characteristic rules, and (iv) the approximate OLAP querying by using OLAP operations on computed approximation spaces.

### 4.3 New Operators for Approximating Concepts

Using a SELECT-like syntax, we define seven instructions to get approximation spaces, positive and negative regions as well as characteristic and classification rules. The user can then utilize one of the following instructions and providing values for the decision dimension (dimension that serves for classification or characterisation purposes) and the dimension member (target value).

- SELECT LOWER decision\_dimension = target\_value
- SELECT UPPER decision\_dimension = target\_value
- SELECT BOUNDARY\_REGION decision\_dimension = target\_value
- SELECT CHARACTERISTIC\_RULES decision\_dimension = target\_value
- SELECT CLASSIFICATION\_RULES decision\_dimension = target\_value
- SELECT POSITIVE\_REGION decision\_dimension
- SELECT NEGATIVE\_REGION decision\_dimension.

The new operators act as filters for existing OLAP cubes by retrieving only cells that belong to the requested approximation. The user can then follow up his exploration by limiting himself to the new generated cube and then applying OLAP and data mining operators on that cube. Such mechanisms bring additional capabilities for analytical processing (e.g., special focus on an approximation space) and an integration of OLAP and data mining facilities.

<sup>4</sup> Remember that in a  $\alpha$ -approximate context, we may have overlapping classes rather than disjoint ones.

## 5 Conclusion

This paper introduces OLAP cube approximation capabilities by integrating notions of the rough set theory into the multidimensional data context. The proposed approach allows the user to get approximate answers in two ways: a tight lower bound which insures that all output elements (cube facts) are in the exact answer or a loose upper bound that may contain a superset of the exact answer. The former mode is useful when the query output is large, and hence allows the user to focus on a reduced set of fully matching tuples. The latter is useful when a query returns an empty or small answer set, and hence allows to relax the query conditions in order to get a larger output. In addition, the proposed approach generates classification and characteristic rules. The classification rules can be exploited to revise the nomenclature proposed by experts by highlighting the similarity that may hold between facts for a given set of dimensions corresponding to a given user's perspective.

Our current work deals with dimensionality reduction and statistical modeling in data warehouses in order to discover useful patterns (e.g., outliers) in data and discard irrelevant dimensions or dimension members. We are also exploring alternatives for a tight coupling of data mining and OLAP operators to study for example the impact of a roll-up operation on an existing rule set.

## Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

## References

1. Chu, W.W., Chen, Q.: A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering* **6** (1994) 738–749
2. Muslea, I.: Machine learning for online query relaxation. In: *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press (2004) 246–255
3. Babcock, B., Chaudhuri, S., Das, G.: Dynamic sample selection for approximate query processing. In: *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ACM Press (2003) 539–550
4. Ganti, V., Lee, M.L., Ramakrishnan, R.: Icicles: Self-tuning samples for approximate query answering. In: *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc. (2000) 176–187
5. Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate query processing using wavelets. *The VLDB Journal* **10** (2001) 199–223
6. Shanmugasundaram, J., Fayyad, U., Bradley, P.S.: Compressed data cubes for olap aggregate query approximation on continuous dimensions. In: *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM Press (1999) 223–232

7. Ambite, J.L., Shahabi, C., Schmidt, R.R., Philpot, A.: Fast approximate evaluation of olap queries for integrated statistical data. In: Proceedings of the First National Conference on Digital Government Research. (2001)
8. Vitter, J.S., Wang, M.: Approximate computation of multidimensional aggregates of sparse data using wavelets. In: Proceeding of the SIGMOD'99 Conference. (1999) 193–204
9. Pawlak, Z.: Rough sets. *International Journal of Computer and Information Sciences* (1982) 341–356
10. Pawlak, Z., Grzymala-Busse, J., Slowinski, R., Ziarko, W.: Rough sets. *Commun. ACM* **38** (1995) 88–95
11. Quafafou, M.: alpha-rst: A generalization of rough sets theory. In: Proc. Fifth Int' Workshop on Rough Sets and Soft Computing (RSSC'97). (1997)
12. Naouali, S., Quafafou, M.: Rough sql : Approximation base querying for pragmatic olap. In: Proceedings of the IEEE Int. Conf. on Information and Communication Technologies: from Theory to Applications (ICTTA'2004). (2004)
13. Naouali, S.: Enrichment of Data Warehouses with Knowledge: Application on the Web (in french). PhD thesis, Université de Nantes, France (2004)

# An Extendible Array Based Implementation of Relational Tables for Multi Dimensional Databases

K.M. Azharul Hasan, Masayuki Kuroda, Naoki Azuma,  
Tatsuo Tsuji, and Ken Higuchi

Graduate School of Engineering, University of Fukui, Fukui shi 910-8507, Japan  
{hasan, kuroda, azuma, tsuji, higuchi}@pear.fuis.fukui-u.ac.jp

**Abstract.** A new implementation scheme for relational tables in multidimensional databases is proposed and evaluated. The scheme implements a relational table by employing a multidimensional array. Using multidimensional arrays provides many advantages, however suffers from some problems. In our scheme, these problems are solved by an efficient scheme of record encoding based on the notion of extendible array. Our scheme exhibits good performance in space and time costs compared with conventional implementation.

## 1 Introduction

The strong need to handle large scale data efficiently has been promoting extensive research themes on organization or implementation schemes for multidimensional array [1][2]. In recent years, on line analytical processing (OLAP) employing multi-dimensional arrays is becoming increasingly important for the analysis of statistical multi dimensional data [3][4]. In this paper, a new implementation scheme for relational tables is proposed and evaluated. The scheme implements a relational table by employing a multidimensional array like in MOLAP (Multidimensional OLAP) systems [4][5]. However, these kinds of multidimensional arrays suffer from the following two problems:

- (1) In general, they are sparse.
- (2) Their sizes are fixed in every dimension; when a new column value is added, array size extension along the dimension is impossible.

In order to solve the problem (2) above, the concept of *extendible array*[6][7] will be employed. An extendible array is extendible in any direction without any relocation of the data already stored. Such advantages make it possible to be applied into wide application area including the above mentioned where necessary array size cannot be predicted and can be varied according to the dynamic environment during operating time of the system.

In this research, an efficient method of encoding records is proposed for relational tables. This encoding is based on the extension history of the underlying (logical) extendible array and the offset value in the subarray. Our scheme will

be called as HORT (History-Offset implementation of Relational Tables). It can be effectively applied not only to the implementation of relational tables administrated by usual RDBMSs, but also applied to multidimensional database systems [8], or data warehouse systems [9].

## 2 Employing Extendible Arrays

In the conventional implementation, each record is placed on secondary storage one by one in the input order. This arrangement suffers from some shortcomings.

- (1) The same column values in different records will have to be stored many times and hence the volume of the database increases rapidly.
- (2) In the retrieval process of records, unless some indexes are prepared, it is necessary to load records in the table sequentially in main memory and check the column value. Therefore retrieval time tends to be long.

The implementation using multidimensional array can be used to overcome problem (2) above. Such an implementation causes further problems:

- (3) Conventional schemes do not support dynamic extension of an array, hence addition of a new column value is impossible if the size of the dimension overflows.
- (4) In ordinary situation, implemented arrays are very sparse.

The concept of extendible array we will explain next will overcome problem (3). It is based upon the index array model presented in [7].

An  $n$  dimensional extendible array  $A$  has a history counter  $h$  and three kinds of auxiliary tables for each extendible dimension  $i (i = 1, \dots, n)$  See Fig. 1. These tables are history table  $H_i$ , address table  $L_i$  and coefficient table  $C_i$ . The history tables memorize extension history  $h$ . If the size of  $A$  is  $[s_n, s_{n-1}, \dots, s_1]$  and the extended dimension is  $i$ , for an extension of  $A$  along dimension  $i$ , contiguous memory area that forms an  $n - 1$  dimensional subarray  $S$  of size  $[s_n, s_{n-1}, \dots, s_{i+1}, s_{i-1}, \dots, s_2, s_1]$  is dynamically allocated. Then the current history counter value is incremented by one, and it is memorized on the history table

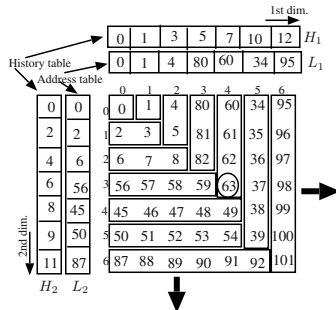


Fig. 1. Realization of 2 dimensional extendible array



$H_i$ , also the first address of  $S$  is held on the address table  $L_i$ . Since  $h$  increases monotonously,  $H_i$  is an ordered set of history values. An element  $\langle i_n, \dots, i_1 \rangle$  in an  $n$  dimensional conventional fixed size array of size  $[s_n, s_{n-1}, \dots, s_1]$  is allocated on memory using an addressing function like this:

$$f(i_n, i_{n-1}, \dots, i_2, i_1) = s_1 s_2 \dots s_{n-1} i_n + s_1 s_2 \dots s_{n-2} i_{n-1} + \dots + s_1 i_2 + i_1.$$

Here, we call  $\langle s_1 s_2 \dots s_{n-1}, s_1 s_2 \dots s_{n-2}, \dots, s_1 \rangle$  as *coefficient vector*. Using these three kinds of auxiliary table, the address of an array element can be computed. as follows. Consider the element  $\langle 4, 3 \rangle$  in Fig. 1. Compare  $H_1[4] = 7$  and  $H_2[3] = 6$ . Since  $H_1[4] > H_2[3]$ , it can be proved that the element  $\langle 4, 3 \rangle$  is involved in the extended subarray  $S$  occupying the address from 60 to 63. The first address of  $S$  is known to be 60, which is stored in  $L_1[4]$ . Since the offset of  $\langle 4, 3 \rangle$  from the first address of  $S$  is 3, the address of the element is 63.

### 3 The HORT Implementation Model

The model that we are going to propose is based on the extendible array explained in the previous section.

**Definition 1 (Logical HORT).** For a relational table  $R$  with  $n$  columns, the corresponding logical structure of HORT is the pair  $(A, M)$ .  $A$  is an  $n$  dimensional extendible array created for  $R$  and  $M$  is the set of mappings. Each  $m_i (1 \leq i \leq n)$  in  $M$  maps the  $i$ -th column values of  $R$  to subscripts of the dimension  $i$  of  $A$ . Hereafter  $A$  will be often called as a *logical extendible array*.

In our HORT technique, we specify an element using the pair of *history value* and *offset value*. Since a history value is unique and has one to one correspondence with the corresponding subarray, the subarray including the specified element of an extendible array can be referred to uniquely by its corresponding history value  $h$ . Moreover the *offset value* (i.e., logical location) of the element is also unique in the subarray. Therefore each element of an  $n$  dimensional extendible array can be referenced to specifying the pair (*history value, offset value*). This reference method of an arbitrary element of an extendible array is very important for our HORT technique. In the coordinate method, if the

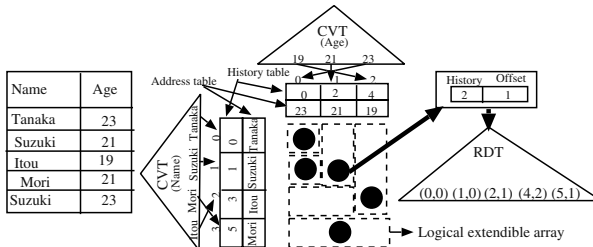


Fig. 2. HORT physical structure

dimension of the extendible array becomes higher, the length of the coordinate becomes longer proportionally. Since an  $n$ -column record can be referenced by its  $n$  dimensional coordinate  $\langle i_n, i_{n-1}, \dots, i_1 \rangle$  in the corresponding multidimensional array, the storage requirement for referencing records become large if the dimension is high. On the contrary, in our *history offset* reference, even if the dimension is high, the size of the reference is fixed in short.

In the HORT logical structure  $(A, M)$ , each mapping  $m_i$  in  $M$  is implemented using a single  $B^+$  tree called CVT(key subscript ConVersion Tree), and the logical extendible array  $A$  is implemented using a single  $B^+$  tree called RDT(Real Data Tree) and a HORT table that is a extension of the three auxiliary tables of an extendible array. An example of HORT physical structure is shown in Fig.2.

**Definition 2 (CVT).**  $CVT_k$  for the  $k$ -th column of an  $n$  columns relational table is defined as a structure of  $B^+$  tree with each distinct column value  $v$  as a key value and its associated data value is subscript  $i$  of the  $k$ -th dimension of the logical extendible array. Hence the entry of the sequence set of the  $B^+$  tree is the pair  $(v, i)$ . The subscript  $i$  references to the corresponding entry of the HORT table that will be defined in the next definition.

**Definition 3 (HT).** HT(HORT Table) corresponds to the auxiliary tables explained in Section 2. It includes the history table and the coefficient table. Note that the address table is void in our HORT physical implementation. Besides these two kinds of information, the information referenced by the subscript in CVT includes a *counter* that counts the number of records with the corresponding column value, and the *column value* itself. If the existing keys are inserted into the CVT, the *counter* field in HT slot is incremented.

HT is arranged in sequential manner according to the insertion order. For example, the column value "Mori" is mapped to the subscript 3 as the insertion order, though in the sequence set of CVT, the key "Mori" is in position 2 due to the property of  $B^+$  tree. Each column value in the record is inserted to the corresponding CVT as a key value. If the key value already exists then the *counter* field in the HT is incremented by one, otherwise the logical extendible array  $A$  is extended by one along the dimension, and a new slot in HT is assigned and initialized.

**Definition 4 (RDT).** The set of the pairs (*history value*, *offset value*) for all of the effective elements in the extendible array are housed as the keys in a  $B^+$  tree called RDT. Here, the effective elements mean the ones that correspond to the records in the relational table. Note that the RDT together with the HTs implements the logical extendible array on storage. We assume that the key occupies the fixed size storage and the *history value* is arranged in front of the *offset value*. Hence the keys are arranged in the order of the history values and keys that have the same history values are arranged consecutively in the sequence set of RDT.

**Definition 5 (HORT).** For an  $n$  columns relational table, its HORT (History-Offset implementation of Relational Tables) implementation is the set of  $n$  CVTs,  $n$  HTs and the RDT.

From the above definition,  $(A, M)$  in Definition 1,  $A$  is mapped to (RDT,  $n$  HTs) and  $M$  is mapped to  $n$  CVTs.

At the end of this section, we summarize that the problems discussed in Section 2 are all resolved in our HORT model. Even if the same column value in different records emerges, the value is stored in the corresponding CVT only once. Moreover, even if the records are so lengthy, they are handled as the set of pairs (*history value, offset value*) in RDT. This greatly reduces the total volume of the database. Hence the problem (1) is resolved in our model. The random addressing mechanism of a record in an extendible array contributes to solve the problem (2); The support of dynamic extension in extendible arrays solves the problem (3), The extension of an extendible array is performed logically in HORT model, and physically only the position information of the effective array elements is stored in RDT, which overcomes the problem (4).

## 4 Operations on HORT

### 4.1 Existence Check of a Record

Let  $r = \langle v_1, v_2, \dots, v_n \rangle$  be a record to be checked. First by searching each CVT, the tuple of the subscripts  $I = \langle CVT_1(v_1), CVT_2(v_2), \dots, CVT_n(v_n) \rangle$  is determined. Here  $CVT_i(v_i)$  denotes the mapped subscript for the column value  $v_i$  in  $CVT_i$ . If at least one of the column values is not found in its corresponding CVT,  $r$  does not exist in  $R$ . Otherwise, the pair of the history value and the offset value  $\langle h, o \rangle$  for  $I$  is determined. Then using the pair as an input key, RDT is searched. If it is found, the record  $r$  exists in  $R$ , otherwise does not exist.

### 4.2 Insertion of a Record

As in Section 4.1, the tuple of the subscripts  $I = \langle CVT_1(v_1), CVT_2(v_2), \dots, CVT_n(v_n) \rangle$  is checked. If every column value is found in its CVT, RDT is searched being  $\langle h, o \rangle$  for  $I$  as a key. If it is not found, the key  $\langle h, o \rangle$  is stored in RDT. If there exist column values that are not found in their CVT's, let  $d_1, d_2, \dots, d_k (1 \leq k \leq n)$  be such dimensions corresponding to these columns in the logical extendible array. For each  $d_i (1 \leq i \leq k)$  the followings are performed.

- (a) The column value  $v_{d_i}$  is stored in  $CVT_{d_i}$
- (b) The logical extendible array is extended by one along the dimension  $d_i$ . Namely incrementing the history counter value by one, the value is memorized on the next slot of  $HT_{d_i}$ . Then, the coefficient vector is computed and memorized.

When the above (a) and (b) are completed on every dimension  $d_i (1 \leq i \leq k)$ , the key  $\langle h, o \rangle$  for the element  $I$  is computed and stored in RDT.

### 4.3 Deletion of a Record

The specified record  $r = \langle v_1, v_2, \dots, v_n \rangle$  is searched according to the procedure in Section 4.1. If it is found in RDT, the corresponding key value  $\langle h, o \rangle$  is

deleted. Then the maintenance of CVTs and HTs are done; for each  $CVT_i$ , the history counter field of  $HT_i[CVT_i(v_i)]$  is decremented by one. If the counter value becomes 0,  $v_i$  is deleted from  $CVT_i$ .

#### 4.4 Retrieval of Records

Here retrieval of records means to search records, some of whose column values are specified. Let the dimensions corresponding to these columns be  $d_1, d_2, \dots, d_k$  and their values be  $v_{d_1}, v_{d_2}, \dots, v_{d_k}$ . Let  $h_{d_1}, h_{d_2}, \dots, h_{d_k}$  be the history values that correspond to the subscripts  $CVT_{d_1}(v_{d_1}), \dots, CVT_{d_k}(v_{d_k})$  and the maximum history value be  $h_{max} = \max(h_{d_1}, h_{d_2}, \dots, h_{d_k})$ . The subarray corresponding to  $h_{max}$  is named as the *principal subarray* in the following.

*Key-subscripts conversion and matching keys.* Let  $\langle h, o \rangle$  be a key in the sequence set of RDT. From  $\langle h, o \rangle$ , the corresponding tuple of the subscripts  $\langle i_n, i_{n-1}, \dots, i_1 \rangle$  of the logical extendible array can be uniquely computed. In order to do fast computation, a one dimensional array HA is prepared on main memory. For the history value  $h$  of the allocated subarray,  $HA[h]$  memorizes its dimension  $d$  and the subscript value  $i_d$ . For the specified key  $\langle h, o \rangle$ , by looking  $HA[h]$ , we can locate the corresponding principal subarray very quickly. The subscripts of the dimensions other than  $d$  can be simply computed by repeated divisions by knowing the coefficient vector stored in  $HT_d[i_d]$ . If the computed subscripts  $i_{d_1}, i_{d_2}, \dots, i_{d_k}$  are equal to  $CVT_{d_1}(v_{d_1}), CVT_{d_2}(v_{d_2}), \dots, CVT_{d_k}(v_{d_k})$  respectively,  $\langle h, o \rangle$  proves to match the retrieval condition.

*Retrieving offsets from RDT.* To retrieve from RDT we defined the following flags depending on the requirements of retrieval.

GTEQUAL: Returns the smallest key that is greater than or equal to the specified key if it exists otherwise returns NO.

NEXT: Returns the NEXT key of previously returned key in the sequence set.

**Naive method (Method 1).** The search starts working from the root of the RDT with key value  $\langle h_{max}, 0 \rangle$  with the flag GTEQUAL. After that, sequential search is performed with the flag NEXT in the sequence set until required number of records is found or the end of sequence set is reached; the number is memorized in *counter* field of *tbl\_entry* in Definition 3. The keys matching the retrieval condition are included in the retrieval results.

**Sophisticated method (Method 2).** The subarrays that belong to the dimensions of the known subscripts are not the candidates to searching except the principal subarray. Method 2 searches the candidate subarrays only; i.e., the principal subarray and the subarrays that have history values greater than  $h_{max}$  and do not belong to the known dimensions. In this method, the candidate offset values to be searched in a candidate subarray are determined exactly. Let  $[s_{n-1}, s_{n-2}, \dots, s_1]$  be the size of a candidate subarray.

*Example 1.* Consider a 5 dimensional subarray of size  $[2, 5, 3, 4, 4]$  and assume that the keys in RDT to be searched are corresponding to the elements  $\langle$

$x_5, a_4, x_3, a_2, x_1 >$  of the subarray, where  $x_1 = 0, 1, 2, 3$ ,  $a_2 = 2$ ,  $x_3 = 0, 1, 2$ ,  $a_4 = 3$  and  $x_5 = 0, 1$ . There are total  $s_1 \times s_2 \times s_3 = 24$  offset values  $\{152, 153, 154, 155\}$ ,  $\{168, 169, 170, 171\}$ ,  $\{184, 185, 186, 187\}$ ,  $\{392, 393, 394, 395\}$ ,  $\{408, 409, 410, 411\}$ ,  $\{424, 425, 426, 427\}$  that are the candidates for searching.

*Target periods.* The candidate offsets are organized periodically. The period is determined by the dimensions of the known subscripts to be searched. Let total  $k$  dimensions  $(d_k, d_{k-1}, \dots, d_1)$  be known where  $d_1 < d_2 < \dots < d_{k-1} < d_k$ . The period  $p_{d_i}$  ( $1 \leq i \leq k$ ) is given by  $p_{d_i} = \prod_{j=1}^{d_i} s_j$ . Of them,  $tr_{d_i} = \prod_{j=1}^{d_i-1} s_j$  offsets are the *target range* of offsets which are the candidates for searching and the offsets  $ntr_{d_i} = (s_{d_i} - 1) \prod_{j=1}^{d_i-1} s_j$  are the *non target range* of offsets which are not the candidates of the subscripts. The total number of candidate offsets are given by  $\prod_{j=1}^n s_j (j \neq d_i, i = 1, \dots, k)$ . In Example 1, total 24 offsets are candidates with two periods  $p_2$  and  $p_4$ . The period for the known subscript  $a_2$  is  $p_2 = s_1 \times s_2 = s_1 + s_1 \times (s_2 - 1)$ , where  $tr_2 = s_1$  and  $ntr_2 = s_1 \times (s_2 - 1)$ .

*Distribution of records.* Consider  $n$  columns relation  $R$  with  $NR$  records. The records in a relation  $R$  are assumed to be uniformly distributed in the corresponding logical extendible array. If the number of distinct keys of the  $i$ -th column is  $L_i$  then the duplicate factor of the  $i$ -th column is defined by  $dp_i = NR/L_i$ . The total logical space of the corresponding logical extendible array required for  $R$  is  $S = (NR/dp_1) \times (NR/dp_2) \times \dots \times (NR/dp_n)$ . Hence the density of records in the logical extendible array is  $\rho = NR/S$ .

*Searching Scheme in Subarrays.* Let  $k$  dimensions  $(d_k, d_{k-1}, \dots, d_1)$  are known where  $d_1 < d_2 < \dots < d_{k-1} < d_k$ . Assume that the number of nodes needed for storing  $tr_{d_1}$  and  $ntr_{d_1}$  be  $tr\_nd_{d_1}$  and  $ntr\_nd_{d_1}$  respectively and the height of RDT not including the sequence set nodes is  $h_R$ .  $tr\_nd_{d_1}$  is determined by  $tr\_nd_{d_1} = \lceil (tr_{d_1} \times \rho) / kn \rceil$ , and so on, where  $kn$  is the average number of keys in a node of RDT.

If  $h_R < ntr\_nd_{d_q}$  then it will be faster to traverse from the root node of RDT to reach the first target node of the next period  $p_{d_q}$  than continue to search  $tr\_nd_{d_q}$  nodes in the period rather than to search the sequence set sequentially like in the naive method (Method 1). Among  $k$  dimensions that are known, the dimension  $d_q$  ( $1 \leq q \leq k$ ) is determined as the one that satisfies  $ntr\_nd_{d_{q-1}} \leq h_R \leq ntr\_nd_{d_q}$ . A set called *partial sequence set* is prepared that contains  $tr\_nd_{d_q}$  nodes which are to be searched sequentially by the flag NEXT. The search starts in the subarray from the root node of RDT with the flag GTEQUAL followed by sequential search in the partial sequence set by the flag NEXT.

## 5 Related Works

The well known grid file principle was first proposed in [10]. If the data distribution is less uniform in grid file implementation then the ratio of grid regions to number of buckets increases and the expansion of the directory approaches to an exponential rate. Most of the grid region corresponds to empty blocks

and the problem is magnified by the number of dimensions. The introduction of multi level grid file [11] improves the situation for non uniform data distribution but the directory expansion is still exponential. Buckets should contain records themselves like in traditional databases, and not a scalar value as in HORT. This causes considerable storage compensation. Moreover the range of the column values is statically predetermined and any column value beyond the range can not be handled in grid file. Many literatures discuss region preserving schemes for multidimensional analysis. Among of them, some schemes such as Gamma and Theta partitioning [12], Z ordering [13], or Hilbert curve ordering [14] partition whole multidimensional array into non overlapped subarrays like in HORT. But they are based on static partitioning of the space and dynamic extension of the length of dimension is impossible. Moreover the element access methods in these schemes are rather complicated than our HORT access method. Another important technique is the bitmap indexing scheme [15]. Its access method is sufficiently fast, but the important drawback is that the huge overhead of the bitmap storage would be caused when the density of records in the array is low. Moreover, dynamic treatment of bitmap storage is necessary, when a new column value emerges beyond the assumed range; this cost would be high.

## 6 Analytical Evaluation

We ignore the detail cost model here only the experimental results are explained. The parameters are described in the appendix. All lengths or sizes of storage areas are in bytes.

### Storage cost comparison between conventional and HORT implementations.

In the conventional implementation of  $R$ , each of the records in  $R$  occupies consecutive storage hence the storage cost is  $n \times kl \times NR$ . In the following, we call the conventional implementation as CI and the HORT implementation as HI. We evaluate the storage costs of both implementations for  $n = 10$ . In general the cost of HI is,  $n \times \text{cost of } CVT + \text{cost of } RDT + n \times \text{cost of } HORT \text{ table}$ . Among these three costs, the cost of CVT and cost of HORT table are heavily dependent on  $dp$ . The cost of a CVT is  $LN_C \times XC + NLN_C \times XCL$  and the cost of a HORT table is  $L \times tl$ . Note that the cost of RDT is independent of  $dp$ ,  $n$  and  $kl$ . Hence if  $dp$  is very small, then HI has large cost but when  $dp$  increases being  $NR$  fixed, then  $LN_C$ ,  $NLN_C$  and  $L$  decrease and hence the cost becomes smaller. If the  $i$ -th key column is unique key column it is quite obvious that the cost of CVT increases and hence the cost of HI. In this case, an index is usually constructed for the key column, which costs large storage in CI as well.

### Retrieval cost comparison between conventional and HORT implementations.

We computed the retrieval cost of both implementations for  $kl = 16$  and  $n = 6$ . HI always accesses fewer pages comparing with CI.

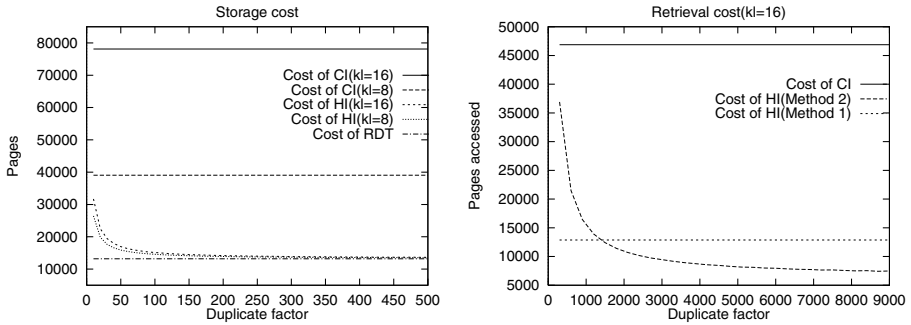


Fig. 3. Storage and retrieval cost analysis with varying duplicate factor

In method 1 of HI,  $\lceil (1 - (1/2)^n \times LN_R)/n \rceil$  nodes are accessed in RDT which corresponds to the known dimension of the extendible array. Hence if the number of records for a subarray increase (i.e.,  $dp$  increase) then Method 1 accesses more nodes which is avoided in Method 2. The cost of Method 2 is higher than Method 1 for small  $dp$  but when  $dp$  increases the cost decreases. This is because, if  $\rho$  increases (i.e.  $dp$  increases) then the number of nodes for the *non target range* (see Section 4.4) of offsets increase and when it becomes greater than  $h_R$ , method 2 scans  $h_R$  nodes instead of accessing the nodes of *non target range* of offsets. Hence for large  $dp$  Method 2 of HI has better performance than Method 1. Throughout the retrieval process we used  $kl = 16$ . For larger  $kl$ , the impact in HI is negligible because  $kl$  has no effect in RDT but  $kl$  has immense effect on CI.

Fig. 3 shows the storage and retrieval cost for CI and HI with varying  $dp$ .

## 7 Conclusions

In this paper, we introduced a new implementation scheme of relational tables for multidimensional database illustrating insertion, deletion and query operations. We developed a cost model for computing the storage cost and retrieval cost. Using these models, we computed and compared with the conventional implementation of relational tables. It is proved that if the duplicate factor is not negligible then HI has improved storage and retrieval performance than the conventional implementation of relational table.

## References

1. S. Sarawagi, M. Stonebraker: Efficient organization of large multidimensional arrays. Proc. of ICDE (1994) 328-336.
2. Y. Zaho, P. M. Deshpande, J. F. Naughton: An array based algorithm for simultaneous multidimensional aggregates. ACM SIGMOD (1997) 159-170.
3. H. Gupta, V. Harinarayan, A. Rajaraman, J. D. Ullman: Index selection for OLAP. Proc. of ICDE (1997) 208-219.

4. F. Buccafuri, D. Rosaci, D. Sacca: Compressed data cube for fast OLAP applications. Proc. of DaWak (1999) 65-77.
5. T. Tsuji, A. Isshiki, T. Hochin, K. Higuchi: An implementation scheme of multi-dimensional arrays for MOLAP. Proc. of DEXA Workshop (2002) 773-778.
6. A. L. Rosenberg: Allocating storage for extendible arrays. JACM, vol, 21 (1974) 652-670.
7. E. J. Otoo, T. H. Merrett: A storage scheme for extendible arrays. Computing, vol. 31 (1983) 1-9.
8. T. B. Pedersen, C. S. Jensen: Multidimensional database technology. IEEE Computer, 34(12) (2001) 40-46.
9. J. Marcus: Index structures for data warehouses. Springer (2002).
10. J. Nievergelt, H. Hinterberger, K. C. Sevic: The grid file: An adaptable, symmetric multikey file structure. ACM Transactions on Database Systems, 9(1):38-71 (1984).
11. K. Y. Whang, R. Krishnamurthy: The multilevel grid file: A dynamic hierarchical multidimensional file structure. Proceedings of DASFAA, (1991) 449-459.
12. R. Orlandic, J. Lukaszuk: A class of region preserving space transformations for indexing high dimensional data. Journal of Computer Science 1 (1), (2005) 89-97.
13. R. H. Gutting: An introduction to spatial database systems. International Journal on Very Large Data Base, 9(4) (1994) 357-399.
14. J. K. Lawder, P. J. H. King: Querying multi-dimensional data indexed using the Hilbert space-filling curve. ACM SIGMOD Record, 30(1), (2001) 19-24.
15. C. Chan, Y. Ioannidis: Bitmap index design and evaluation. ACM SIGMOID (1998) 355-366.

## Appendix: Parameters

(1) Parameters related to the relational table  $R$ :

$NR$  : Total number of records in  $R$ ;  $n$ : Number of columns in  $R$ ;

$L_i$ : Number of distinct column values of the  $i$ -th column;

$dp_i$ : Duplicate factor of the  $i$ -th column;

$\rho$ : Density of records in the logical extendible array;  $kl$ : Length of a column value of  $R$ ;  $P$ : Disk page size.

(2) Parameters for HORT:

$f$ : Average fan out from a node;  $kn$ : Average number of keys in a node;

$pl$ : Length of a pointer;  $tl$  : Length of an element of HORT table.

*Parameters for RDT:*

$klr$ : Length of a key (i.e. length of history value and offset value) of RDT;

$LN_R$ : Number of leaf nodes for RDT;  $NLN_R$ : Number of non leaf nodes for RDT;

$h_R$  : The height of the RDT not including sequence set nodes.

*Parameters for CVT:*

$XC$  : Average size of a non leaf node;  $XCL$ : Average size of a leaf node for CVT;

$LN_C$ : Number of leaf nodes for CVT;  $NLN_C$ : Number of non leaf nodes for CVT.

(3) Assumptions:

(i) The column length of  $R$  is the same for all columns i.e.  $LR = n \times kl$ .

(ii) The duplicate factor  $dp_i$  of the  $i$ -th column is same for all  $i$ . Hence we write  $dp$  for  $dp_i$  and  $L$  for  $L_i$ .

The values of some parameters that we assumed are as follows:

$klr=4+8$ ,  $kl=8$  and  $16$ ,  $n=10$  and  $6$ ,  $NR=1000000$ ,  $f=102$ ,  $P=2048$ ,  $pl=8$ .



# Nearest Neighbor Search on Vertically Partitioned High-Dimensional Data

Evangelos Dellis, Bernhard Seeger, and Akrivi Vlachou

Department of Mathematics and Computer Science, University of Marburg,  
Hans-Meerwein-Straße, 35032 Germany  
{dellis, seeger, vlachou}@mathematik.uni-marburg.de

**Abstract.** In this paper, we present a new approach to indexing multidimensional data that is particularly suitable for the efficient incremental processing of nearest neighbor queries. The basic idea is to use index-stripping that vertically splits the data space into multiple low- and medium-dimensional data spaces. The data from each of these lower-dimensional subspaces is organized by using a standard multi-dimensional index structure. In order to perform incremental NN-queries on top of index-stripping efficiently, we first develop an algorithm for merging the results received from the underlying indexes. Then, an accurate cost model relying on a power law is presented that determines an appropriate number of indexes. Moreover, we consider the problem of dimension assignment, where each dimension is assigned to a lower-dimensional subspace, such that the cost of nearest neighbor queries is minimized. Our experiments confirm the validity of our cost model and evaluate the performance of our approach.

## 1 Introduction

During the last decades, an increasing number of applications, such as medical imaging, molecular biology, multimedia and computer aided design, have emerged where a large amount of high dimensional data points have to be processed. Instead of exact match queries, these applications require an efficient support for similarity queries. Among these similarity queries, the  $k$ -nearest neighbor query ( $k$ -NN query), which delivers the  $k$  nearest points to a query point, is of particular importance for the applications mentioned above.

Different algorithms have been proposed [14, 16, 19] for supporting  $k$ -NN queries on multidimensional index-structures, like R-trees. Multidimensional indexing has extensively been examined in the past, see [12] for a survey of various techniques. The performance of these algorithms highly depends on the quality of the underlying index. The most serious problem of multi-dimensional index-structures is that they are not able to cope with a high number of dimensions. This disadvantage can be alleviated by applying dimensionality reduction techniques [3]. However, the reduced dimensionality still remains too high for most common index-structures like R-trees. After the high effort put into implementing R-trees in commercial database management systems (DBMS), it seems that their application scope is unfortunately quite limited to low-dimensional data.

In this paper, we revisit the problem of employing R-trees (or other multidimensional index-structures) for supporting  $k$ -NN queries in an iterative fashion. Our fundamental assumption is that high-dimensional real-world data sets are not independently and uniformly distributed. If this assumption does not hold, the problem is not manageable due to the well-known effects in high-dimensional space, see for example [24]. The validity of our assumptions allows the transformation of high-dimensional data into a lower dimensional space. There are, however, two serious problems with this approach. First, the dimensionality of the transformed data might still be too high, in order to manage the data with an R-tree efficiently. Second, this approach is feasible only when the most important dimensions are globally valid independent from the position of the query point. In our new approach, we address both of these problems by partitioning the data vertically and then indexing the (low-dimensional) data of each partition separately.

Our basic architecture is outlined in Fig. 1. As mentioned above, the essential technique of our approach is to partition the data points vertically among different R-trees such that each dimension is assigned to one of them. Thus, the Cartesian product of the subspaces yields the original data space again. An incremental query is processed by running a local incremental query for each of the indexes concurrently.

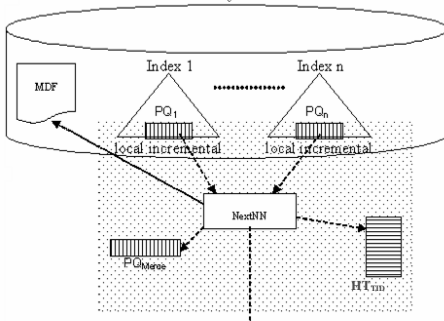


Fig. 1. Overview

(MDF) on disk, where each point is accessible via a tuple identifier ( $TID$ ). Additionally, a priority queue  $PQ_{Merge}$  and a hash table  $HT_{TID}$  are necessary for an efficient iterative processing of  $k$ -NN queries. The key problem that arises from our architecture is to determine an appropriate number of indexes for supporting  $k$ -NN queries efficiently. Thereafter, an effective assignment of the dimensions is indispensable. These problems are addressed in our paper. Moreover, we present different iterative algorithms for  $k$ -NN queries that employ the different indexes dynamically. Our approach is independent from the specific index-structure. We decide to use R-trees in the rest of the paper, simply because these structures are generally available and perform well in practice. For the sake of simplicity, we assume that the Euclidean metric  $L_2$  is used as distance function throughout our paper.

The remaining paper is structured in the following way. In the next Section we discuss previous work. In Section 3, we present a multi-step nearest neighbor algorithm that dynamically exploits the information of internal nodes of the R-tree. Thereafter in Section 4, we provide a cost model relying on a power law and we propose a formula for computing the number of indexes. In Section 5 we present a dimension assignment algorithm. Our results of our experiments are presented in Section 6, and finally, we conclude in Section 7.

## 2 Related Work

Our work is closely related to the nearest neighbor algorithms that have been proposed in the context of multidimensional indexing [14, 16, 19]. Some of the proposals are dedicated to the problem of  $k$ -NN queries when  $k$  is known in advance, whereas others deal with the problem of distance scanning [14] also termed distance browsing [16] and incremental ranking where  $k$  is unknown and the query stops on demand. There are a large number of multidimensional index-structures that have been tailor-cut to supporting  $k$ -NN queries efficiently. Different filter-and-refine algorithms for  $k$ -NN queries were first presented in [18, 21]. These approaches, also known as Global Dimensionality Reduction (GDR) methods, are unable to handle datasets that are not globally correlated. In [7], a structure which is called Local Dimensionality Reduction (LDR) is presented that also exploits multiple indexes, one for each cluster. Unfortunately this strategy is not able to detect all the correlated clusters effectively, because it does not consider correlation nor dependency between the dimensions. Recently, there have been approaches where the high-dimensional points are mapped into a one-dimensional space [27, 28].

The origin of our work starts from [2] where a quite similar work has been proposed for supporting range queries on high-dimensional data. Based on a uniform cost model, the authors first present a formula for the optimal number of multi-dimensional indexes. Two techniques are then presented for assigning dimensions to indexes. The first one simply follows a round-robin strategy, whereas the other exploits the selectivity of the different dimensions such that the dimension with the highest selectivity is assigned to the first index and so on. The assignment strategy offers some deficiencies which makes them inadequate for  $k$ -NN queries. First, this strategy assumes dimensions being independently from each other. This may obviously lead to suboptimal assignments. Second, this strategy is even not applicable to high-dimensional data and  $k$ -NN queries. Since dimensions generally have the same domain,  $k$ -NN queries are symmetric in all dimensions and consequently, the selectivity is constant for all dimensions. Our approach is different from [2] as a new dimension assignment strategy is derived from the fractal dimension [1] of the data set. Moreover, instead of intersecting local results, a merge algorithm outputs the results in an iterative fashion.

Accurate cost models are important to our algorithms for decision making. We are primarily interested in cost models for estimating the cost for  $k$ -NN queries. There are many different cost models that have been developed for R-trees, such that [23]. The first models [24] were developed for uniformly and independently distributed data. Later, these models were extended to cope with non-uniform data. However, independency of dimensions is still assumed for the majority of the models, but unfortunately not satisfied for real data distributions, in general. The usage of the fractal dimension [17, 4] has lead to more accurate cost model since multidimensional data tend to behave in a fractal manner. Our approach employs such a cost model for automatically setting important parameters.

In parallel to the work on incremental ranking in the indexing community, there have been independently studies on a quite similar topic, so-called top- $k$  queries in the area of multimedia databases [10, 11].

### 3 Incremental Nearest Neighbor Algorithms

In this section, we present three iterative algorithms for  $k$ -NN queries that comply with our architecture. Let  $n$  be the number of indexes, where each of them is responsible for a lower-dimensional subspace of the original data space. A global queue  $PQ_{Merge}$  is used for merging local results of the indexes. Since an index only delivers partial points, i.e. points from the subspaces, the multidimensional file ( $MDF$ ) has to be accessed to obtain the entire point. We have developed the following three different incremental nearest neighbor algorithms, called Best-Fit, TA-Index and TA-Index<sup>+</sup>.

**Best-Fit:** This algorithm is a straightforward adoption of the classical incremental algorithm [14, 16] for more than one index. During the initialization step of the query, the roots of all  $n$  indexes are inserted into  $PQ_{Merge}$ . In each iteration step, the algorithm pops the top element from  $PQ_{Merge}$ . If the top element is an index entry, the entry is expanded, i.e., the referenced page is read from the corresponding index and its entries (points) are added to  $PQ_{Merge}$ . If the top element is a partial point that is the first time on top, the entire point is read from  $MDF$  and inserted into  $PQ_{Merge}$ . In addition, we store its tuple identifier in the hash table  $HT_{TID}$ . We actually use  $HT_{TID}$  to check whether a partial point is the first time on top. If the top element is an entire point, we deliver the point as the next neighbor to the user. Note that partial points that appear more than once on top can be safely discarded.

**TA-Index:** This algorithm performs similar to TA [11], but supports arbitrary NN queries due to the fact that the indexes are able to deliver the data in an appropriate order. TA-Index performs different to Best-Fit as a local incremental NN query runs for every index concurrently. The partial points, which are delivered as the results of these local queries, are merged by inserting them into  $PQ_{Merge}$ . The algorithm performs similar to Best-Fit as partial points that appear on top are replaced by their entire points. In analogy to TA, we keep a partial threshold  $min_i$  for the  $i^{th}$  index where  $min_i$  is the distance between the last partial result of the  $i^{th}$  index and the query point of the NN query. An entire point from  $PQ_{Merge}$  is returned as the next nearest neighbor if its distance is below the global threshold that is defined as the squared sum of the partial thresholds. The faster the partial thresholds increase the higher the chance that the top element of  $PQ_{Merge}$  will become a result.

**TA-Index<sup>+</sup>:** This algorithm is an optimized version of TA-Index as it additionally exploits the internal index entries to speed up the increase of the partial thresholds. Different to TA-Index is that the incremental NN queries on the local indexes are assumed to deliver not only the data points, but also the internal entries that are visited during processing. Note

---

```

nextNearestNeighbor()


---


while ( $\sqrt{\min_1^2 + \dots + \min_n^2} < L_2(q, PQ_{Merge}.top())$ ) do
    ind = activityScheduler.next();
    cand = incNNplus(ind).next();
     $\min_{ind} = L_{2,ind}(q, cand)$ ;
    if ((cand is a point) and not
         $HT_{TID}.contains(cand.TID)$ ) then
        obj =  $MDF.get(cand.TID)$ ;
         $PQ_{Merge}.push(obj)$ ;
         $HT_{TID}.add(obj.TID)$ ;
    return  $PQ_{Merge}.pop()$ ;

```

---

**Fig. 2.** Algorithm TA-Index+

that the distance of the delivered entries and points is continuously increasing. Therefore, the distances to index entries can also be used for updating  $min_i$  without threatening correctness. The incremental step of the algorithm is outlined in Fig. 2.

In order to improve the performance of the algorithms, it is important to reduce the number of candidates that have to be examined. The number largely depends on how fast the termination inequality (condition of the while-loop in Fig. 2) is satisfied in our algorithms. Therefore, we develop effective strategies for computing a so called *activity schedule* that determines in which order the indexes are visited. The goal of the activity schedule is a fast enlargement of the sum of the local thresholds  $min_i$ . The naive strategy is to use a round-robin schedule. This strategy however does not give priority to those indexes which are more beneficial for increasing the local thresholds. Similar to [13], we develop a heuristic based on the assumption that indexes that were beneficial in the recent past will also be beneficial in the near future.

Our activity schedule takes into account the current value of the partial threshold  $min_i$  as well as the number of candidates  $c_i$  an index has already delivered. The more candidates an index has delivered the less priority should be given to the index. Each time a candidate is received from an index, we increment the counter  $c_i$ . If the candidate has been already examined (by another index), we give an extra penalty to the  $i^{th}$  index by incrementing  $c_i$  once again. Our activity schedule then gives the control to the local incremental algorithm with maximum  $min_i/c_i$ . Note that our heuristic prevents undesirable situations where one index keeps control for a very long period.

In case of TA-Index<sup>+</sup> the distances to index entries influences not only the thresholds, but also the activity schedule. The internal nodes help to exploit the R-trees dynamically in a more beneficial way such that the number of examined candidates is reduced. This is an important advantage against the common TA [11].

The cost of our algorithm is expressed by a weighted sum of I/O and CPU cost. The I/O cost is basically expressed in the number of page accesses.  $IO_{index}$  refers to the number of accesses incurred from the local query processing, whereas  $IO_{cand}$  is the number of page accesses required to read the candidates from *MDF*. Note that  $IO_{cand}$  is equal to the total number of candidates if no buffer is used. The CPU cost consists of two parts. First, the processing cost for the temporary data structures like  $PQ_{Merge}$  and  $HT_{TD}$ . The more crucial part of the CPU cost might arise from the distance calculations.

## 4 A Cost Model Based on Power Law

We decided to use the fractal dimension as the basis for our cost model. The reason is twofold. First, there are many experimental studies, see [17], showing that multidimensional data sets obey a power law. Second, cost models for high-dimensional indexing, which go beyond the uniformity assumption, often employ a power law [4, 17]. Therefore, the fractal dimension seems to be a natural choice for designing a cost model.

In the following, we show that there is a strong relationship between the fractal dimension and the performance of nearest neighbor queries, especially in the case of multi-step algorithms. Let us assume that the data does not follow a uniform and in-

dependent distribution and that the query points follow the same distribution as the data points.

Given a set of points  $P$  with finite cardinality  $N$  embedded in a unit hypercube of dimension  $d$  and its fractal (correlation) dimension  $D_2$ , the average number of neighbors  $\overline{nb}(r, D_2)$  of a point within a region of regular shape and radius  $r$  obeys the power law:

$$\overline{nb}(r, D_2) = (N - 1) \cdot Vol(r)^{\frac{D_2}{d}},$$

where  $Vol(r)$  is the volume of a shape (e.g. cube, sphere) of radius  $r$ , see [17]. The average Euclidean distance  $r$  of a data point to its  $k$ -th nearest neighbor is given by:

$$r = \frac{\left(\Gamma\left(1 + \frac{d}{2}\right)\right)^{\frac{1}{d}}}{\sqrt{\pi}} \cdot \left(\frac{k}{N - 1}\right)^{\frac{1}{D_2}}.$$

Since the ratio  $\frac{k}{N - 1}$  is always smaller than 1, an interpretation of the above formula reveals that a lower fractal dimension leads to a smaller average distance. A multi-step algorithm [21], where one index is used for indexing a subspace, is called optimal if exactly the points within this region are examined as candidates. The number of candidates that have to be examined by the optimal multi-step algorithm depends on the fractal dimension  $D'_2$  of the subspace, i.e. partial fractal dimension [25], with embedded dimensionality  $d'$ , as shown in the following formula:

$$\overline{nb}(r, D'_2) = (N - 1) \cdot Vol(r)^{\frac{D'_2}{d'}} = (N - 1) \cdot \frac{(\sqrt{\pi} \cdot r)^{D'_2}}{\Gamma\left(1 + \frac{d'}{2}\right)^{\frac{D'_2}{d'}}}.$$

The above formula shows that a higher fractal dimension produces fewer candidates for the same radius. Thus, it is obvious that the performance of an optimal multi-step algorithm depends on the fractal dimension of the subspace that is indexed. In our multi-step algorithm, where more than one indexes are used, a region  $r_i$  of each index  $i$ , ( $1 \leq i \leq n$ ), such that  $r = \sqrt{\sum_{1 \leq i \leq n} r_i^2}$  is dynamically exploited.

The average number of candidates for an index on a subspace with finite cardinality  $N$ , embedded in a unit hypercube of dimension  $d_i$  and fractal dimension  $D_{2,i}$ , is given by:

$$\overline{nb}(r_i, D_{2,i}) = (N - 1) \cdot Vol(r)^{\frac{D_{2,i}}{d_i}} = (N - 1) \cdot \frac{(\sqrt{\pi} \cdot r)^{D_{2,i}}}{\Gamma\left(1 + \frac{d_i}{2}\right)^{\frac{D_{2,i}}{d_i}}}.$$

The total number of candidates is:  $\overline{nb}(r) = \sum_{1 \leq i \leq n} \overline{nb}(r_i, D_{2,i})$ .

In the above formula we include the duplicates that are eliminated during the process of our algorithm, but the number of duplicates is of minor importance for our cost

model. For the evaluation of this formula some assumptions are necessary. The goal of our cost model and dimension assignment algorithm is to establish a good performance for all indexes. Therefore, we assume that  $d_i = \frac{d}{n} = d'$  and  $D_{2,i} = D'_2$  for each index  $i$ . If this assumptions hold it is expected that the indexes are equally exploited based on the activity scheduler, thus  $r_i = \frac{r}{\sqrt{n}} = r'$ .

The I/O cost can be one page access per candidate in the worst case. As observed in [7] and as confirmed by our experiments this assumption is overly pessimistic. We, therefore, assume the I/O cost of the candidates to be the number of candidates divided by 2:

$$A_{cand}(r) = \frac{n}{2} \cdot (N-1) \cdot \frac{(\sqrt{\pi} \cdot r)^{D'_2}}{\Gamma\left(1 + \frac{d'}{2}\right)^{\frac{D'_2}{d'}}$$

The I/O cost of processing a local  $k$  nearest neighbor query is equal to the cost of the equivalent range query of radius  $r_i$ . The expected number of page accesses for a range query can be determined by multiplying the access probability with the number of data pages  $\frac{N}{C_{eff}}$ , where  $C_{eff}$  the effective capacity. Based on the previous assumptions, the total cost of processing the local queries is:

$$A_{index}(r) = n \cdot \frac{N}{C_{eff}} \cdot \left( \sum_{0 \leq j \leq d'} \binom{d'}{j} \cdot \left( \left(1 - \frac{1}{C_{eff}}\right)^{D'_2} \sqrt{\frac{C_{eff}}{N-1}} \right)^j \cdot \frac{\sqrt{\pi}^{d'-j}}{\Gamma\left(\frac{d'-j}{2} + 1\right)} \cdot (r')^{d'-j} \right)^{\frac{D'_2}{d'}}$$

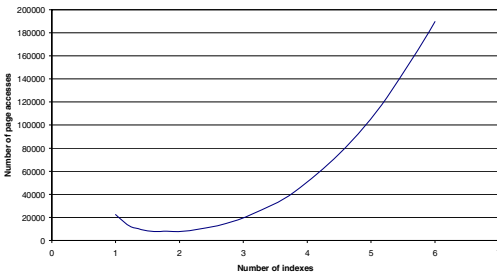


Fig. 3. Total Cost of k-NN query

fractal dimension  $D_2$  of the dataset is 10.56. We evaluate the parameter  $k$  by setting the expected average selectivity of a query as 0.01%. There is a clear minimum in  $n=2$ .

In our cost model we assume that the indexes currently maintain partial data points, whereas it might also be possible to keep the entire data points (in the leaves). This offers the advantage that  $MDF$  is not necessary anymore and therefore,  $A_{cand} = 0$ . However, the cost for processing the local incremental queries increase since, in case of the R-tree, the capacity of the leaves is reduced by a factor of  $n$ .

The cost for processing a local nearest neighbor query within the lower-dimensional subspaces will increase with an increasing partial fractal dimension. Therefore our cost model defines the number of indexes where the sum of all I/O is minimized.

Fig. 3 shows the total cost over  $n$  indexes in a database consisting of 1,312,173 Fourier points in a 16-dimensional data space. The

## 5 Dimension Assignment Algorithm

In this section, we sketch our algorithm for assigning  $d$  dimensions to  $n$  indexes,  $n < d$ . Based on the observation that subspaces with low fractal dimension produces more candidates, we follow two basic goals. First, each of the  $n$  indexes should have a high fractal dimension. Second, the fractal dimension should be equal for all indexes and therefore the indexes should perform similarly. These goals conform to the assumptions of our cost model and enhance the applicability of the cost model.

In general, an optimal solution of the dimension assignment problem is not feasible because of the large number of attributes. Instead, we employ a simple greedy heuristic that is inexpensive to compute while still producing near-optimal assignments.

A greedy algorithm is developed that starts with assigning the  $n$  attributes with the highest partial fractal dimension [25] to the indexes, where every index receives exactly one attribute. In the next iterations, an attribute, which has yet not being assigned to an index, is assigned to the index with minimum fractal dimension. The index receives the attribute that maximize the fractal dimension of the index. Notice that not all indexes have necessarily the same number of attributes.

Our algorithm is outlined in fig. 4. The function  $D_2(S)$  calculates the fractal dimension of a subspace  $S$ .  $A_i$  represents the  $i^{\text{th}}$  attribute;  $L$  refers to the set of unassigned attributes, and  $da_i$  is the set of attributes assigned to the  $i^{\text{th}}$  index. The algorithm calls  $O(d^2)$  times the box-counting algorithm [1] which calculates the fractal dimension in linear time w.r.t. the size of the dataset.

---

```

assignDimensions()


---


L = {A1, ..., Ad};
for (i=1, ..., n) do
    Amax = argmaxA ∈ L { D2(A) };
    dadim = Amax ∪ dadim;
    L = L - {Amax};
while (L is not empty) do
    dim = argmin1 ≤ j ≤ n D2(daj);
    Amax = argmaxA ∈ L D2(A ∪ dadim);
    dadim = Amax ∪ dadim;
    L = L - {Amax};

```

---

**Fig. 4.** Our Greedy Algorithm

## 6 Experimental Validation

For the experimental evaluation of our approach, indexes were created on two data sets from real-world applications. We set the page size to 4K and each dimension was represented by a double precision real number. Both data sets contain features of 68,040 photo images extracted from the Corel Draw database. The first dataset contains 16-dimensional points of image features, while the second dataset contains 32-dimensional points that represent color histograms. In all our experiments, we examined incremental  $k$ -NN queries where the distribution of the query points follows the distribution of the data points. All our measurements were averaged over 100 queries.

In order to illustrate the accuracy of our cost model, we show in Fig. 5 the estimated page accesses and the actual page accesses measured during the execution of 10-NN queries by Best-Fit. The plot shows the I/O cost for both data sets where the fractal dimensions of the 16-dimensional and 32-dimensional datasets are 7.5 and 7.1, respectively. Note that the relative error is less than 20% in our experiments. Fur-



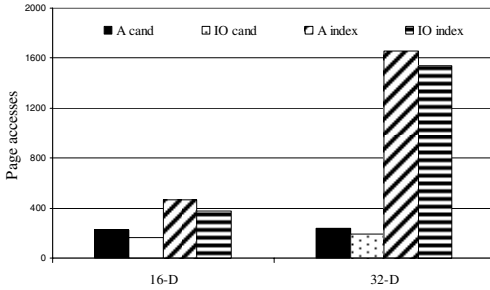


Fig. 5. Accuracy of our Cost Model

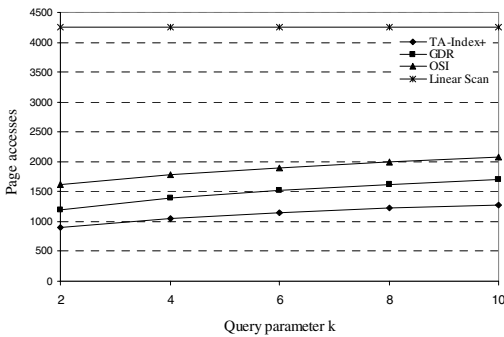


Fig. 6. Comparative Study

contains four curves reporting the I/O as a function of  $k$ , where  $k$  denotes the desired number of neighbors. As illustrated, TA-Index<sup>+</sup> constantly outperforms the other techniques in our experiment.

## 7 Conclusions

The indexing problem for high-dimensional data is among the practically relevant research problems where solutions are difficult to achieve. The processing of nearest neighbor queries becomes only meaningful for high-dimensional data if we assume that there are high correlations in the data. We provide a new approach for querying vertically partitioned high-dimensional data. Three new algorithms for processing incremental nearest neighbor queries have been presented. In order to provide fast query processing, we addressed the problem of dimension assignment and provided approximate solutions based on the usage of the fractal dimension. An accurate cost model relying on a power law is presented. The results obtained by the experiments with real data sets consistently gave evidence that our technique is also indeed beneficial in practical scenarios.

Furthermore, TA-Index<sup>+</sup> produces fewer page accesses, due to the usage of an activity schedule which is not considered by the cost model.

In the second experiment, we compare the following four techniques: TA-Index<sup>+</sup>, Original Space Indexing (OSI), Global Dimensionality Reduction (GDR) and Linear Scan. We examined the I/O cost of  $k$ -NN queries for the 32-dimensional dataset. For TA-Index<sup>+</sup>, we created two 16-dimensional R-trees as it was required from our cost model. Each of the corresponding subspaces has a fractal dimension of almost 3.75, resulting in well-performing R-trees. For the GDR method, we used the optimal multi-step algorithm on a 16-dimensional R-tree. The results of our experiment are plotted in Fig. 6. The plot contains

## References

1. Belussi A., Faloutsos C. Self-Spatial Join Selectivity Estimation Using Fractal Concepts. *ACM Transactions on Information Systems (TOIS)*, Volume 16, Number 2, pp 161-201, 1998
2. Berchtold S., Böhm C., Keim D., Kriegel H-P., Xu X. Optimal multidimensional query processing using tree striping. *Int. Conf. on Data Warehousing and Knowledge Discovery*, 2000, pp 244-257.
3. Bingham E., Mannila H. Random projection in dimensionality reduction: applications to image and text data. *Int. Conf. on Knowledge discovery and data mining, ACM SIGKDD*, 2001, pp 245-250.
4. Böhm C. A cost model for query processing in high dimensional data spaces. *ACM Transactions on Database Systems (TODS)*, Volume 25, Number 2, pp 129-178, 2000
5. Böhm C., Berchtold S., Keim D. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, Vol. 33, pp 322-373, 2001
6. Böhm K., Mlivoncic M., Schek H.-J., Weber R. Fast Evaluation Techniques for Complex Similarity Queries. *Int. Conf. on Very Large Databases (VLDB)*, 2001, pp 211-220.
7. Chakrabarti K. and Mehrotra S. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. *Int. Conf. on Very Large Databases (VLDB)*, 2000, pp 89-100
8. Chaudhuri S., Gravano L. Evaluating Top-k Selection Queries. *Int. Conf. on Very Large Databases (VLDB)*, 1999, pp 397-410.
9. Ciaccia P., Patella M., Zezula P. Processing complex similarity queries with distance-based access methods. *Int. Conf. Extending Database Technology (EDBT)*, 1998, pp 9-23.
10. Fagin R., Kumar R., Sivakumar D. Efficient similarity search and classification via rank aggregation. *ACM Symp. on Principles of Database Systems*, 2003, pp 301-312.
11. Fagin R., Lotem A., Naor M. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, Volume 66, Number 4, pages 614-656, 2003
12. Gaede V., and Gunther O. Multidimensional Access Methods. *ACM Computing Surveys*, Volume 30, pp 170-231, 1998
13. Güntzer U., Balkler W-T., Kiessling W. Optimizing multi-feature queries in image databases. *Int. Conf. on Very Large Databases (VLDB)*, 2000, pp 419-428.
14. Henrich A. A Distance Scan Algorithm for Spatial Access Structures. *ACM-GIS*, pp 136-143, 1994
15. Hinneburg A., Aggarwal C., Keim D. What Is the Nearest Neighbor in High Dimensional Spaces? *Int. Conf. on Very Large Databases (VLDB)*, 2000, pp 506-515.
16. Hjaltonson G. R., Samet H. Ranking in Spatial Databases. *Advances in Spatial Databases*, *Int. Symp. on Large Spatial Databases, (SSD)*, LNCS 951, 1995, pp 83-95.
17. Korn F., Pagel B-U., Faloutsos C. On the "Dimensionality Curse" and the "Self-Similarity Blessing". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, No. 1, 2001.
18. Korn F., Sidiropoulos N., Faloutsos C., Siegel E., Protopapas Z. Fast nearest neighbor search in medical image databases. *Int. Conf. on Very Large Databases (VLDB)*, 1996, pp 215-226.
19. Pagel B-U., Korn F., Faloutsos C. Deflating the Dimensionality Curse Using Multiple Fractal Dimensions. *Int. Conf. on Data Engineering (ICDE)*, 2000, pp 589-598.
20. Roussopoulos N., Kelley S., Vincent F. Nearest neighbor queries. *ACM SIGMOD*, 1995, pp 71-79.

21. Seidl T., Kriegel H-P. Optimal Multi-Step k-Nearest Neighbor Search. ACM SIGMOD, 1998, pp 154-165.
22. Tao Y., Faloutsos C., Papadias D. The power-method: a comprehensive estimation technique for multi-dimensional queries. ACM CIKM, Information and Knowledge Management, 2003, pp 83-90.
23. Tao Y., Zhang J., Papadias D., Mamoulis N. An Efficient Cost Model for Optimization of Nearest Neighbor Search in Low and Medium Dimensional Spaces. IEEE TKDE, Vol. 16, No. 10, 2004.
24. Theodoridis Y., Sellis T. A Model for the Prediction of R-tree Performance. ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1996, pp 161-171.
25. Traina C. Jr., Traina A. J. M., Wu L., Faloutsos C. Fast Feature Selection Using Fractal Dimension. SBBD 2000, p.p. 158-171.
26. Weber R., Schek H-J., Blott S. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. Int. Conf. Very Large Databases (VLDB), 1998, pp 194-205.
27. Yu C., Bressan S., Ooi B. C., and Tan K-L. Querying high-dimensional data in single-dimensional space. The VLDB Journal, Vol. 13, pp 105-119, 2004.
28. Yu C., Ooi B. C., Tan K-L., and Jagadish H. V. Indexing the distance: an efficient method to KNN processing. Int. Conf. Very Large Databases (VLDB), 2001, pp 421-430.

# A Machine Learning Approach to Identifying Database Sessions Using Unlabeled Data

Qingsong Yao, Xiangji Huang, and Aijun An\*

York University, Toronto M3J 1P3 Canada  
{qingsong, jhuang, aan}@cs.yorku.ca

**Abstract.** In this paper, we describe a novel co-training based algorithm for identifying database user sessions from database traces. The algorithm learns to identify positive data (session boundaries) and negative data (non-session boundaries) incrementally by using two methods interactively in several iterations. In each iteration, previous identified positive and negative data are used to build better models, which in turn can label some new data and improve performance of further iterations. We also present experimental results.

## 1 Introduction

Database users perform a certain task by submitting a sequence of queries. We call this sequence of queries a database user session. Session identification from a database workload is a prerequisite step for task-oriented session analysis that allows us to discover high-level patterns in user queries, and provides useful insight into database user behavior. Previous session identification methods, such as the timeout and statistical language modeling ( $n$ -gram) based methods [5,8], have some limitations. For example, it is hard to find suitable threshold for the timeout method and a large training data is required for the  $n$ -gram method.

In this paper, we propose a novel co-training based session identification method. The algorithm labels positive data (session boundaries) and negative data (non-session boundaries) incrementally by using the timeout and  $n$ -gram methods interactively in several iterations. In each iteration, we use previously identified positive and negative data to build better timeout and  $n$ -gram models, which in turn can label some new data, and improve performance of further iterations. The algorithm does not require any training data. Our experiment shows that this method overcomes the limitation of previous methods and can achieve a comparable performance .

The rest of the paper is organized as follows. In Section 2, we discuss the related research. We describe our co-training algorithm and parameter selection in Section 3. In Section 4, we describe the data set and experimental results. Finally, we conclude the paper in Section 5.

---

\* This work is supported by research grants from Communications and Information Technology Ontario and the Natural Sciences and Engineering Research Council of Canada.

## 2 Background and Related Research

In this paper, we assume that queries within a user session use the same database connection, and there is no interleave between two sessions of a connection. This rule usually holds since a user obtains a database connection before performing a task, and releases it until the task is finished, even in a connection sharing environment. Then, the task of identifying sessions is to learn session boundaries from the request log of a database connection. The most common and simplest method for session identification is *timeout* where a session shift is identified between two requests if the time interval between the two requests is more than a pre-defined threshold. This method suffers from the problem that it is difficult to set the time threshold since it may be significantly different for different applications or users. A survey of different session detection methods can be found in [5].

In [5], an  $n$ -gram statistical language modeling based session detection method was proposed. The method first builds an  $n$ -gram model from a set of *training data*. Then, the method assigns a value for each request in the testing data. The value of a request  $r_m$  is the empirical entropy of the request history sequence  $s\{r_1, \dots, r_m\}$ , which is defined as  $-\frac{1}{m} \log_2 p(s)$ , where  $p(s)$  is the probability of sequence  $s$  estimated from the  $n$ -gram model, defined as  $\prod_{i=1}^m P(r_m | r_{m-n+1} \dots r_{m-1})$ . With the method, a session boundary could be placed between two requests when the change in entropy between two requests passes a threshold. The method was demonstrated to be more effective than the timeout and two other session detection methods for discovering interesting association rules in a Web mining domain. Recently, we applied the method to identifying database sessions [8]. An important factor that affects the performance of the algorithm is to choose suitable parameters, such as the  $n$ -gram order and entropy threshold. In the application, we used some domain knowledge to generate some training data by manually identifying all or part of the session boundaries. These training data were used to adjust the parameters needed by the method. The experiments showed that the method achieved better performance than the *timeout* method. However, the performance may become worse when such domain knowledge is not available.

*Co-training* is a machine learning technique that was originally proposed by Blum and Mitchell [1]. It assumes that the description of each example can be partitioned into two distinct views. The presence of two distinct views suggests strategies in which two learning algorithms are trained separately on each view, and then each algorithm's predictions on new unlabeled examples are used to enlarge the training set of the other. There are a number of studies that explore the potentials of the co-training method in recent years. Craven [3] uses co-training to extract knowledge from the WWW. The result shows that it can reduce the classification error by a factor of two using only unlabeled data. In [6], Kititchnko and Matwin apply co-training to e-mail classification, and they show that the performance of co-training depends on the learning method it uses. For example, naive Bayesian classifiers perform poorly while support vector machines work well. In [4], Goldman and Zhou use two different supervised learning algorithms to label data for each other. Chan, Koprinska, and Poon [2] suggest to randomly split a single feature space into two different feature sets for the co-training algorithm.

### 3 Session Identification with Co-training

In this section, we present our co-training learning algorithm. The algorithm is an unsupervised algorithm that does not depend on any training data. Instead, it uses the timeout method and the un-supervised  $n$ -gram method together to learn session boundaries incrementally. We assume that a sequence of requests that belong to a database connection is already obtained. The request sequence corresponds to a sequence of sessions. Each user request contains two features: the time interval with the previous request and the request history sequence. These two features are not completely correlated, which enables us to use the idea of *co-training* to identify sessions. We train the timeout model and the  $n$ -gram model<sup>1</sup> from the testing data. These models can help to label data independently. The new labeled data will help to build better models, which in turn results in more labeled data.

In our method, each user request is described by the following parameters: ID ( $id$ ), starting time ( $stime$ ), ending time ( $etime$ ), time interval from the previous request ( $tvalue$ ), entropy value ( $evalue$ ), entropy changes ( $dvalue$ ), and  $status$ , which are illustrated in Table 1. Given a sequence of queries, we first classify the queries into different classes [8]. Then each query in the sequence is replaced with the  $id$  of the query class. Whether a request is a session boundary depends on the value of  $status$ . Initially, all requests are unlabeled, and their  $status$  is unknown (0). In each iteration, the status of a request may be changed by either a timeout model or an  $n$ -gram model. When a request is identified as a session boundary by a method, the  $status$  value is increased by a confidence value. The confidence value is between 0.8 and 1, and represents the prediction accuracy of positive or negative samples. Finally, we can identify all sessions according to the request status.

**Table 1.** User request entry

Name	Meaning
ID ( $id$ )	the id of the request
Starting time ( $stime$ )	the time when server received the request
Ending time ( $etime$ )	the time when server finished processing the request
Time interval ( $tvalue$ )	$tvalue[i] = stime[i] - etime[i - 1]$
Entropy value ( $evalue$ )	the entropy value of current request
Entropy difference ( $dvalue$ )	the relative entropy changes, $dvalue[i] = \frac{evalue[i] - evalue[i-1]}{evalue[i-1]}$
Status ( $status$ )	if $status > 0$ , and it is a session boundary, if $status < 0$ , and it is non-boundary, other unknown.

We use  $F$ -measure as the performance measurement metric in this paper. Suppose we know the true session boundaries in the test sequence, then  $F$ -measure is defined as  $\frac{2 * precision * recall}{precision + recall}$ , where  $precision$  is defined as the ratio of the correctly detected session boundaries to the total estimated boundaries, and the  $recall$  is the hit-rate, that is, the ratio of the correctly detected session boundaries to the total number of true boundaries. A higher  $F$ -measure value means a better overall performance.

<sup>1</sup> Unlike previous algorithms, the  $n$ -gram model is built from the testing data itself instead of using another training data.

### 3.1 Co-training Algorithm

Our co-training algorithm contains two steps: *training step*, and *separating step*. The training step contains several iterations. In each iteration, previously labeled data (i.e., the requests with  $status \neq 0$ ) is used to learn better timeout and  $n$ -gram models, which is used to obtain more labeled data in the next iteration. In the separating step, the previously learned labeled data can be used to build semi-supervised  $n$ -gram model. The model is used to separate the log into sessions according to certain strategy.

The training step is listed in Algorithm 1. The algorithm reads user requests from a log file. In each iteration, it uses the previous labeled data to train a new  $n$ -gram model (the previously labeled positive data can separate a sequence into small sequences, and these small sequences can be used to build the  $n$ -gram model, which is more accurate than the model built from the un-separated data). Then, we estimate the number of requests to be labeled in line 5, where  $pc_t[i]$  and  $nc_t[i]$  are the numbers of positive and negative data, respectively, to be labeled by the timeout method, and  $pc_n[i]$  and  $nc_n[i]$  are the values for the  $n$ -gram method. In [1], the authors use constant values as the number of data to be labeled, which is 1 and 3 for positive and negative data respectively. However, we believe that the numbers should vary with respect to the test data and should be based on the trade-off between *precision* and *recall*. A small number leads to a better prediction precision, but a large number can achieve a good recall. We decide to make the total numbers to be learned dynamically change with regard to different test data. The number is controlled by the input parameters which will be discussed in next section. Once the total numbers are determined, we assign the numbers for each iteration. The strategy is to make the number for the timeout method decreases when the number of iteration increases, and make that of  $n$ -gram method increases. The reason is that the prediction accuracy for the timeout method decreases as the iteration increases, but the accuracy for the  $n$ -gram method increases since a better model is obtained.

Both timeout and  $n$ -gram methods may make wrong estimations, either false positives or false negatives. We should allow this wrong estimation be adjusted in further iterations. Thus, we assign a confidence value for each method in every iteration (in line 6). The value range is between 0.8 to 1.0. If a method identifies a request as a positive data in an iteration, the status is increased with the confidence value, otherwise the value is decreased when it is identified as a negative data. The timeout method has good prediction precision at early iterations, and the accuracy becomes worse in later iterations in order to get high coverage. However, the  $n$ -gram method becomes better in later iteration since the previous identified data can improve the  $n$ -gram model. As a result, the confidence value sequence of the timeout method ( $conf_n[i]$  in Algorithm 2) is assigned decreasingly, but that of  $n$ -gram value,  $conf_n[i]$ , is assigned increasingly.

Our next question is how to choose threshold values. There are totally 4 values to be estimated, where  $low_t[i]$ , and  $high_t[i]$  are for timeout, and  $low_n[i]$ , and  $high_n[i]$  are for  $n$ -gram. If the timeout threshold value of a request is larger than  $high_t[i]$ , we will increase its *status* with a confidence value  $conf_t[i]$  (in line 10, and 11), and so on. The method we use to choose threshold value is the histogram technique. We build/update a timeout value histogram and  $n$ -gram entropy change histogram, referred as  $hist_t$  and  $hist_n$  in line 7,8. The histogram contains several pre-defined buckets, and each request

**Algorithm 1** training( $log, p_{order}, p_{iter}, p_{len}, p_{pos}, p_{neg}, p_{ngram}$ )**Input:** a user request log file  $log$ , and a set of co-training parameters**Output:** user request array  $req$ 


---

```

1: read requests from log file  $data$ , and create a request array  $req$ 
2: initialize every item in  $req$ 
3: for  $i = 0$  to  $p_{iter}$  do
4:   build/rebuild a  $n$ -gram model based on the log data, and update  $evaluate, dvalue$  for each request
5:   estimate the number of data to be learned:  $pc_t[i]$  and  $nc_t[i]$  (for timeout),  $pc_n[i]$  and  $nc_n[i]$  (for  $n$ -gram)
6:   estimate prediction confidence:  $conf_t[i], conf_n[i]$ 
7:   update timeout histogram  $hist_t$ , select threshold value  $low_t[i]$ , and  $high_t[i]$ 
8:   update  $n$ -gram histogram  $hist_n$ , select threshold value  $low_n[i]$ , and  $high_n[i]$ 
9:   for all  $r[j] \in req$  do
10:    if  $tvalue[j] > high_t[i]$  then
11:       $status[j] = status[j] + conf_t[i]$ 
12:    end if
13:    if  $tvalue[j] < low_t[i]$  then
14:       $status[j] = status[j] - conf_t[i]$ 
15:    end if
16:    if  $dvalue[j] > high_n[i]$  then
17:       $status[j] = status[j] + conf_n[i]$ 
18:    end if
19:    if  $dvalue[j] < low_n[i]$  then
20:       $status[j] = status[j] - conf_n[i]$ 
21:    end if
22:  end for
23: end for
24: return  $req$ 

```

---

**Algorithm 2** separating( $req, p_{order}, p_{iter}, p_{len}, p_{pos}, p_{neg}, p_{ngram}$ )**Input:** user request array  $req$ , and a set of co-training parameters**Output:** a set of sessions

---

```

1: rebuild  $n$ -gram model with previous labelled data, and update  $evaluate, dvalue$  for each request
2: update entropy histogram:  $hist_n$ 
3: choose two threshold value:  $low_n$ , and  $high_n$  according to  $hist_n$ 
4: for all request  $r[j] \in data$  do
5:   if  $dvalue[j] > high_n$  then
6:      $status[j] = status[j] + 1.0$ 
7:   end if
8:   if  $dvalue[j] < low_n$  then
9:      $status[j] = status[j] - 1.0$ 
10:  end if
11: end for
12: generate sessions by placing boundary points on positive data

```

---

must be one of the buckets according to its timeout or  $n$ -gram value. Each bucket entry contains following information: the low bound ( $v_{low}$ ), the high bound ( $v_{high}$ ), the number of unknown, positive and negative data. In each iteration, we update the histograms and choose a proper upper and lower threshold values to label data. Figure 1 and Figure 2 are examples of histograms after several iterations, where each value in x-axis is the low bound value of a bucket. The three series illustrate the distribution of the labeled/unlabelled data, and which is used to choose the thresholds. For example, suppose we want to identify 5 new positive data for the timeout method, we can choose a  $v_{low}$  value from  $hist_t$  to allow at least 5 new data with larger timeout value.

After *training* step, the status of some requests are updated. However, there is still some requests have *unknown* status. We will update them in the *separating* step. The algorithm is illustrated in Algorithm 2. We first rebuild a semi-supervised  $n$ -gram model from the request array, and update request entropy values and histogram correspond-



ingly. Then, we choose a final low and high threshold value for  $n$ -gram histogram. The *status* of each request is updated according to the threshold values. Finally, we separate sessions according to request status.

### 3.2 An Example

In this section, we use an example to show how co-training algorithm works on a data set. The data set contains 441 requests belonging to 56 true sessions. In the training step, 59 requests labeled positively, and 36 requests negatively. Table 2 shows the log file after training. We observe that request 31 has a long time interval (24.713s), which leads to a large positive confidence (4.5) and a positive status. However, request 25 has a small entropy change (-0.529), and it has a negative confidence (1.0) and a negative status. A large number of requests are identified as *unknown*. However, such information is enough to train a good  $n$ -gram model in the separating step.

**Table 2.** User request log after applying co-training processing

Pos. ID	Starting Time	Ending Time	Time Interval	Entropy Value	Entropy Change	Status
1 30	15:06:41.052	15:06:41.183	0.000	3.182	-1.0	<b>-10.0</b>
2 9	15:06:42.827	15:06:43.987	2.644	1.735	-0.455	0.00
3 10	15:06:44.777	15:06:44.912	0.790	1.265	-0.271	0.00
4 20	15:06:45.525	15:06:45.823	0.613	1.245	-0.016	0.00
			....			
8 31	15:07:21.800	15:07:21.853	<b>24.713</b>	1.606	0.349	<b>4.50</b>
			....			
56 25	15:09:37.957	15:09:38.124	2.414	1.442	<b>-0.529</b>	<b>-1.00</b>

We generated the intermediate result for each iteration, and calculated the precision and recall values. The result is shown in Table 3. At the beginning, the prediction precision is very high, which means the proposed algorithm can find some session boundaries correctly. In later iterations, the precision begins to drop. However, the coverage (recall) increases. The final result has a *0.91* accuracy in term of *F-measure*.

**Table 3.** Intermediate result in each iteration

Method	Iteration No.	Estimated	Matched	Precision	Recall	F-Measure
timeout	1	15	15	1.00	0.25	0.41
$n$ -gram	1	15	15	1.00	0.25	0.41
timeout	2	15	15	1.00	0.25	0.41
$n$ -gram	2	15	15	1.00	0.25	0.41
timeout	3	16	16	1.00	0.27	0.43
$n$ -gram	3	20	20	1.00	0.34	0.51
timeout	4	31	30	0.97	0.51	0.67
$n$ -gram	4	43	40	0.93	0.68	0.78
timeout	5	49	42	0.86	0.71	0.78
$n$ -gram	5	61	52	0.85	0.88	0.87
Final	N/A	60	53	0.88	0.95	<b>0.91</b>

Figures 1 and 2 show the timeout and  $n$ -gram histograms after the training step. From the figure, we observe that a large amount of unlabeled data is within the middle range, some positive data is on the right side, and some negative data is on the left side.

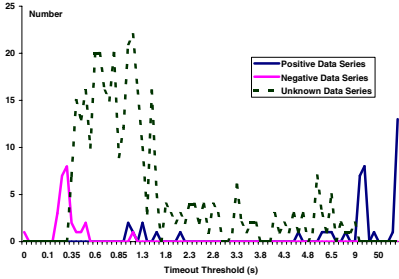


Fig. 1. Timeout threshold histogram after training step

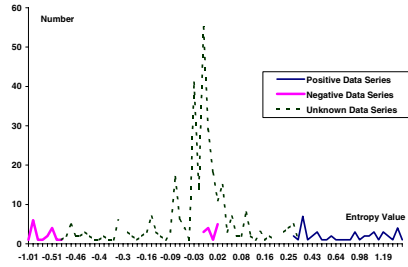


Fig. 2.  $n$ -gram entropy change histogram after training step

### 3.3 Co-training Parameters

In this section, we discuss the parameters and how to choose optimal values. Table 4 shows the parameters used in our algorithm and their value ranges. We assume that a parameter is independent of the others. To study the influence of these parameters, we did 63,000 experiments with different parameters on our testing data (see table 5 for details). These results are used to compare the performance under different parameters.

Table 4. Co-training Parameters

Name	Meaning	Range
$p_{order}$	$n$ -gram order	1-10
$p_{iter}$	Iteration number	1-20
$p_{len}$	Average session length	vary in applications
$p_{pos}$	Ratio of positive data to be identified	0.0 - 1.0
$p_{neg}$	Ratio of negative data to be identified	0.0 - 1.0
$p_{ngram}$	Ratio of data to be identified by $n$ -gram method	0.0 - 1.0

In  $n$ -gram modeling, it is assumed that the probability of a word only depends on its at most previous  $n-1$  words, where  $n$  is the *order* of the model. Parameter  $p_{order}$  represents the  $n$ -gram order, it is usually between 1 and 10. The choice of  $n$  is based on a trade-off between detail and reliability, and will be dependent on the quantity of available training data. In the field of speech recognition, for the quantities of language model training data typically available at present, 3-gram models strike the best balance between precision and robustness, although interest is growing in moving to 4-gram models and beyond. In Figure 3, we plot the best performance in term of  $F$ -measure for different representative  $p_{order}$ . We observe that the 3-gram and 4-gram have the better performance than the others.

Parameter  $p_{iter}$  is the iteration number. In [1], the authors choose 30 as the iteration number. We conduct experiments with different iterations ranging from 2 to 20, and compare the performance. The result is shown in Figure 4. We observe that small iteration number, such as 2, has poor performance. Also, a large number ( $>8$ ) may not have benefit, and an iteration number between 5 and 8 is suitable for our algorithm.

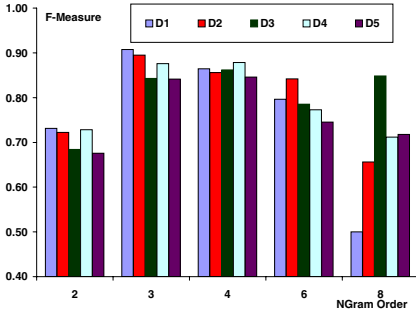


Fig. 3. Performance comparison of different  $n$ -gram number

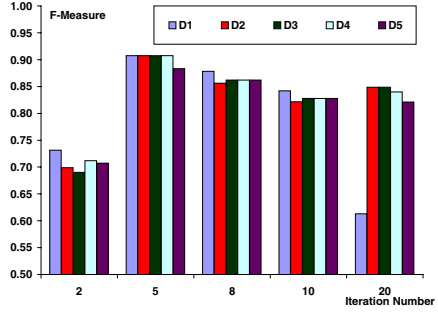


Fig. 4. Performance comparison of different iteration number

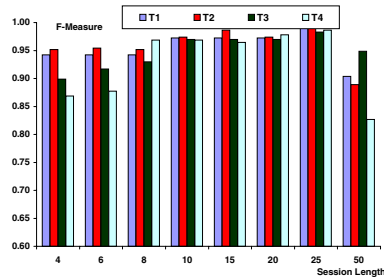
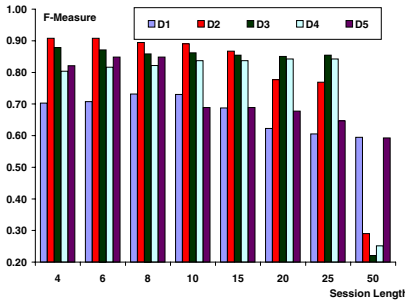


Fig. 5. Performance comparison of different session length

The next parameter  $p_{len}$  is an estimated average session length. The purpose of this parameter is to estimate how many sessions in a testing data, and the number of positive data as well. In our OLTP testing data, the value is between 6 to 10, and it is 26 for TPCC benchmark data (see Table 5 for details). We conduct experiments on different  $p_{len}$  values, ranging from 4 to 50. The result, illustrated in Figure 5, shows that the performance of our algorithm is not very sensitive to this value. Even we makes wrong estimation (such as 15 for OLTP, 8 for TPCC), the algorithm can still achieve good performance.

Parameters  $p_{pos}$  and  $p_{neg}$  determine how many positive/negative data to be learned in training step. The number of labeled data to be learned by the two methods are certain ratios ( $p_{pos}$ , and  $p_{neg}$ ) to the session number. Given a log file contains  $n$  requests,  $\frac{n}{p_{len}} * p_{pos}$  positive labels, and  $(n - \frac{n}{p_{len}}) * p_{neg}$  negative labels will be learned in training step. Since many requests are non-session (negative) boundary, and learning large negative data does not help the algorithm too much. We decide to make  $p_{neg}$  smaller than 0.5. We compare the performance under different positive ratios and negative ratios in Figure 6. The result shows that an optimal value of  $p_{pos}$  is between 0.5 and 0.9, and the algorithm is not sensitive to  $p_{neg}$  value.

The last parameter  $p_{ngram}$  is the ratio of labeled data to be learned by the  $n$ -gram method. The timeout method and  $n$ -gram method have different accuracies for differ-

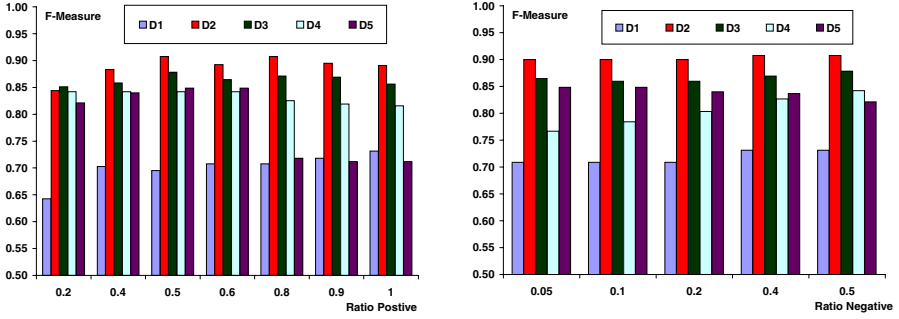


Fig. 6. Performance comparison of different positive/negative ratio

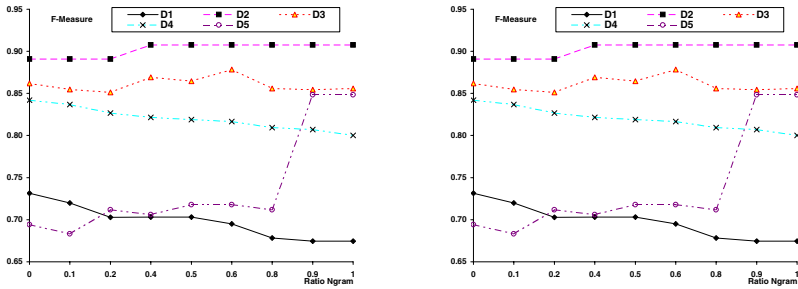


Fig. 7. Performance Comparison of different  $n$ -gram ratio

ent testing data. For some testing data, the timeout method is more accurate than the  $n$ -gram method, thus we will allow it to identify more labeled data. Otherwise, the  $n$ -gram method will identify more labeled data. We use the parameter  $p_{ngram}$  to control the identification ratio between the two methods. Figure 7 shows the result for different  $n$ -gram ratio value. For testing data  $D2$  and  $D5$ , a large value achieves the best performance. However, a small value achieve better performance for  $D1$  and  $D4$ . The result suggests that the optimal value of  $p_{ngram}$  is sensitive to the data set.

## 4 Data Sets and Experimental Results

We collected two data sets to testing our idea. The first one is a clinic OLTP application. The clinic is a private physiotherapy clinic located in Toronto. In each day, the client applications installed in a branch connects to the database and performs different tasks, such as checking-in customer, making appointments, etc. We collect 400M bytes database logs within 10 hour observation time. After preprocessing the log, we obtain 7,244 SQL queries. The queries are classified into 190 categories, and 18 connection log files are obtained. In order to compare performance, we manually separate these log files into sessions. 5 representative logs as the testing data, and the others are the training data for the supervised  $n$ -gram learning and unsupervised  $n$ -gram learning methods proposed in [8]. The training data has 5,450 requests and 789 session.

**Table 5.** Testing Data Set

OLTP Data				TPCC Data		
Name	Requests	Sessions	Length	Name	Requests	Sessions
D1	677	65	10.0	T0	7062	272
D2	441	56	08.0	T1	5064	193
D3	816	118	07.0	T2	40207	1569
D4	1181	153	07.5	T3	51405	2748
D5	452	69	06.6	T4	82852	3227

**Table 6.** Experiment Result for the two data set

Name	OLTP Data						TPCC Data				
	D1	D2	D3	D4	D5	Overall	T1	T2	T3	T4	Overall
Timeout	0.72	0.69	0.63	0.70	0.39	<b>0.63</b>	1.0	1.0	0.72	0.60	<b>0.83</b>
Unsupervised	0.75	0.88	0.89	0.68	0.49	<b>0.73 (16%)</b>	0.92	0.90	0.90	0.82	<b>0.88 (6%)</b>
Supervised	0.77	0.94	0.89	0.84	0.97	<b>0.88 (40%)</b>	0.97	0.97	0.96	0.89	<b>0.95 (14%)</b>
Co-training	0.73	0.91	0.88	0.84	0.85	<b>0.84 (33%)</b>	0.99	0.96	0.94	0.85	<b>0.92 (11%)</b>
Final	0.56	0.87	0.83	0.70	0.63	<b>0.72 (14%)</b>	0.92	0.89	0.94	0.81	<b>0.89 (7%)</b>

The second data set we used is the query log of TPC-C Benchmark [7]. TPC-C models a wholesale supplier managing orders which involves a mix of 5 different sessions operated against a database of 9 tables. Some sessions contain multiple database transactions (such as, Delivery transaction), and some even does not corresponds to a database transaction (such as Stock Level Transaction). Even the business logic of these transaction is simple, the runtime behavior is complex. Thus, a method is need to separated sessions from the query log efficiently. We collect five log files, which are listed in Table 5, where data  $T0$  is used for training data of algorithms proposed in [8]. The TPC-C implementation can use the connection sharing technique to improve performance. In the testing data,  $T0$ ,  $T1$ , and  $T2$  has no connection sharing, but the number of users who share a connection is 20 and 60 in  $T3$  and  $T4$  respectively.

We first compare the performance of timeout, unsupervised, supervised learning proposed in [8], and our co-training algorithm in Table 6. We examine the best performance that can be achieved by these methods. The result shows that our co-training method out-perform than the timeout and unsupervised learning, but it is less than the supervised learning. Then, we choose *3-gram model* with 5 iterations. The session length is 6 for OLTP application, and 22 for TPC-C benchmark, and 0.8, 0.1, and 0.7 are positive ratio, negative ratio and  $n$ -gram ratio. The result shows that the *final* performance is comparable with the best unsupervised learning, and the *average* performance is comparable with the best timeout method. Meanwhile, our method does not need well selected training data. The timeout method performs good when there is no connection sharing (for data  $T1$ , and  $T2$ ). However, its performance decreases when the number of users who share the database connection increase ( $T3$  and  $T4$ ). The reason is that the time interval between sessions will decrease, and that between request will increase when shared users increases.

## 5 Conclusions

We have presented a co-training based session identification algorithm. Unlike previous algorithms, it does not require any training data or any time threshold. The proposed

algorithm is evaluated on two data sets. The result demonstrates that our algorithm can identify sessions effectively from the testing data. One limitation of the algorithm is that the time and space requirement is larger than previous algorithms. We are currently investigating how to choose co-training parameters more effectively since some testing data is sensitive to the training parameters (such as data  $D_5$ ). Also, in Section 3.3, we assume that the parameters are independent, but they may have certain relationship. For example, the session length and the positive ratio together determine the number of positive labels to be learned. We can use this feature to develop a more scalable algorithm, such as adjusting positive ratio to remedy the wrongly estimated session length.

## References

1. A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proc. of the Workshop on Computational Learning Theory*, pages 92–100, 1998.
2. J. Chan, I. Koprinska, and J. Poon. Co-training with a Single Natural Feature Set Applied to Email Classification. In *Proc. of the 9th Australasian Document Computing Symposium*, pages 47–54, 2004.
3. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, 118(1/2):69–113, 2000.
4. S. Goldman and Y. Zhou. Enhancing Supervised Learning with Unlabeled Data. In *Proc. 17th ICML*, pages 327–334, 2000.
5. X. Huang, F. Peng, A. An, and D. Schuurmans. Dynamic web log session identification with statistical language models. *J. of American Soc. for Info. Sci&Tech*, 55(14):1290–1303, 2004.
6. S. Kiritchenko and S. Matwin. Email Classification with Co-training. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*, 2001.
7. TPC-C. Transaction Processing Performance Council (TPC) Benchmark C Standard Specification Revision 5.0, Feb 2001.
8. Q. Yao, A. An, and X. Huang. Finding and analyzing database user sessions. accepted in the 10th international conference on database systems for advanced applications, 2005.

# Hybrid System of Case-Based Reasoning and Neural Network for Symbolic Features

Kwang Hyuk Im, Tae Hyun Kim, and Sang Chan Park

Department of Industrial Engineering,  
Korea Advanced Institute of Science and Technology (KAIST),  
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Korea  
{gunni, imiss}@major.kaist.ac.kr, sangchanpark@kaist.ac.kr

**Abstract.** Case-based reasoning is one of the most frequently used tools in data mining. Though it has been proved to be useful in many problems, it is noted to have shortcomings such as feature weighting problems. In previous research, we proposed a hybrid system of case-based reasoning and neural network. In the system, the feature weights are extracted from the trained neural network, and used to improve retrieval accuracy of case-based reasoning. However, this system has worked best in domains in which all features had numeric values. When the feature values are symbolic, nearest neighbor methods typically resort to much simpler metrics, such as counting the features that match. A more sophisticated treatment of the feature space is required in symbolic domains. We propose another hybrid system of case-based reasoning and neural network, which uses value difference metric (VDM) for symbolic features. The proposed system is validated by datasets in symbolic domains.

## 1 Introduction

Case-based reasoning (CBR) is frequently applied to data mining with various objectives. CBR has common advantages over other learning strategies. It can be directly applied to classification without additional transformation mechanisms and has strength in learning the dynamic behavior of the system over a period of time. Unfortunately, it also has shortcomings to be applied to real world tasks. It suffers from the feature weighting problem. In this framework, similar case retrieval plays an important role, and the  $k$ -nearest neighbor ( $k$ -NN) method or its variants are widely used as the retrieval mechanism. However, the most important assumption of  $k$ -NN is that all of the features presented are equally important, which is not true in many practical applications. When CBR measures the distance between cases, some input features should be treated more important than other features. Many variants of  $k$ -NN have been proposed to assign higher weights to the more relevant features for case retrieval. Though many feature-weighted variants of  $k$ -NN have been reported to improve its retrieval accuracy on some tasks [1-3], few have been used in conjunction with the neural network learning until Shin *et al.* proposed a hybrid approach of neural network (NN) and CBR named as MANN (Memory And Neural Network based learning) [4].

In the hybrid approach of MANN [4], the feature weight set, which is calculated from the trained neural network, plays the core role in connecting both learning strategies and the explanation for prediction, which can be given by obtaining and presenting the most similar examples from the case base. And four feature weighting methods are suggested - Sensitivity, Activity, Saliency, Relevance. Those methods gave large weight value to important features, which improved retrieval accuracy of CBR. The system is designed to process data in domains which all features have numeric values. It uses normalized Euclidean distance to measure similarity of numeric features. In symbolic domains, we must adopt new metric that can measure similarity of symbolic features.

Value Difference Metric (VDM) presented by Stanfill and Waltz takes into account the overall similarity of classification of all instances. For each feature, matrix defining the distance between all possible feature values is derived statistically based on the examples in the training set [5]. We adopted a modified VDM, which has introduced in [6], as the distance measure in symbolic domains. We named the proposed system CANSY (a hybrid system of Case-based reasoning And Neural network for SYmbolic features).

The paper is organized as follows. The MANN, our previous research, is discussed in Section 2.1, and VDM is reviewed in Section 2.2. The overall learning structure of the CANSY, our proposed system, is presented in Section 2.3. In Section 3, illustrative experimental results are presented. We use datasets from the UCI machine learning archive for experiments [7]. We conclude the paper by briefly discussing the limitations of the study and future research directions in Section 4.

## 2 Hybrid System of Case-Based Reasoning and Neural Network Based on VDM

### 2.1 MANN : A Numeric Feature Weighting Algorithm Using a Trained Neural Network

Shin. *et al.* proposed the hybrid approach of case-based reasoning and neural network named as MANN (Memory And Neural Network based learning). This hybrid approach adopts neural network and memory to realize an analogy to the human information processing system. After being trained, the neural network keeps its knowledge in the connection weights among the neurons. The neural network is expected to contain the intrinsic nature of the training dataset completely, and once the network is trained properly the training dataset itself is not utilized any more. However, the thinking process of human brain is apparently aided by the memory (the training dataset in the machine learning case) as well as the connection weights between neurons. In data mining 'memory' is realized in the form of database, which can store, query, and retrieve large amounts of data in a short time. Now database is the fundamental information resource in corporate information systems. It means that, with proper case retrieval methods, we can easily benefit from the abundant database. The approach uses the  $k$ -nearest neighbor ( $k$ -NN) method for case retrieval in CBR settings.  $k$ -NN assumes that each case  $x = \{x_1, x_2, \dots, x_n, x_c\}$  is defined by a set of  $n$  features, where  $x_c$  is  $x$ 's class value (discrete or numeric).



In [4], Given a query  $q$  and a case library  $L$ ,  $k$ -NN retrieves the set  $K$  of  $q$ 's  $k$  most similar (i.e., of least distance) cases in  $L$  and predicts their weighted majority class value as the class value of  $q$ . Distance is defined as

$$D(x, q) = \sqrt{\sum_{f=1}^n W_f \times \text{difference}(x_f, q_f)^2} \tag{1}$$

where  $w_f$  is the parameterized weight value assigned to feature  $f$ , and

$$\text{difference}(x_f, q_f) = \begin{cases} |x_f - q_f| & \text{if } \text{feature } f \text{ is numeric} \\ 0 & \text{if feature } f \text{ is symbolic and } x_f = q_f \\ 1 & \text{if } \text{otherwise} \end{cases} \tag{2}$$

The brief procedure of calculating weights of features,  $w_f$  ( $f=1, \dots, n$ ) are as follows. First, a neural network is trained completely with the given dataset. With the trained neural network and the training data, we calculate the feature weight set according to the methods—Sensitivity, Activity, Saliency, and Relevance. Then, when a new query comes in, we can point out  $k$ -nearest neighbors in the training data based on the feature weight sets. The prediction value of a neural network may also be utilized in conjunction with the neighborhood information. This provides extended information for the query with most similar cases in the database.

The hybrid system of case-based reasoning and neural network improved retrieval accuracy of CBR. This system has worked best in domains in which all features had numeric values, because normalized Euclidean distance was used to compare examples. When the features have symbolic or unordered values (e.g., the letters of the alphabet, which have no natural inter-letter “distance”), it typically resorts to much simpler metrics, such as counting the features that match. Simpler metrics may fail to capture the complexity of the problem domains, and as a result may not perform well. In symbolic domains, a more sophisticated treatment of the feature space is required [6].

**2.2 Value Difference Metric(VDM)**

In domains with symbolic features, the “overlap” metric is usually used, counting the number of features that differ [11]. However, it is observed that the overlap metric gives relatively poor performance in their learning tasks in symbolic feature domains [6]. A new powerful metric for measuring the difference between two instances in domains with the symbolic features was proposed which was called value difference metric (VDM) [5]. VDM takes into account similarity of feature values. The value difference metric is defined by (3).

$$\Delta(X, Y) = \sum_{i=1}^N \delta(x_i, y_i) \tag{3}$$

$$\delta(x_i, y_i) = d(x_i, y_i)w(x_i, y_i) \tag{4}$$

$$d(x_i, y_i) = \sum_{l=1}^n \left| \frac{D(f_i = x_i \cap g = c_l)}{D(f_i = x_i)} - \frac{D(f_i = y_i \cap g = c_l)}{D(f_i = y_i)} \right|^k \tag{5}$$

$$w(x_i, y_i) = \sqrt{\sum_{l=1}^n \left( \frac{D(f_i = x_i \cap g = c_l)}{D(f_i = x_i)} \right)^2} \tag{6}$$

$X$  and  $Y$  are two instances.  $x_i$  and  $y_i$  are values of the  $i^{th}$  feature for  $X$  and  $Y$ .  $N$  is the number of features and  $n$  is the number of classes.  $f_i$  and  $g$  indicate the  $i^{th}$  predicate feature and the class feature, respectively.  $c_l$  is one of possible classes.  $D(condition)$  is the number of instances in a given training dataset which satisfy the condition.

$d(x_i, y_i)$  is a term for measuring the difference overall similarity between feature values  $x_i$  and  $y_i$ . The term  $\frac{D(f_i = x_i \cap g = c_l)}{D(f_i = x_i)}$  is the likelihood that an instance with  $x_i$

of  $i^{th}$  feature value will be classified as class  $c_l$ .  $d(x_i, y_i)$  has a small value if two values give similar likelihoods for all possible classes and this means that two values are similar. Though Stanfill and Waltz used the value of  $k=2$  in their equation, Cost and Salzberg observed that experiments indicated that equally good performance is achieved when  $k=1$ . We also used the value of  $k=1$  for simplicity [6].

$w(x_i, y_i)$  measures the strength with which the  $i^{th}$  feature constrain the values of the class. This measure represents the importance of each feature in classification. In our paper, we remove this term in order to give the same weight to features because the classification information is not given in clustering tasks. Our value difference metric in this paper is

$$\Delta(X, Y) = \sum_{i=1}^N \delta'(x_i, y_i) \tag{7}$$

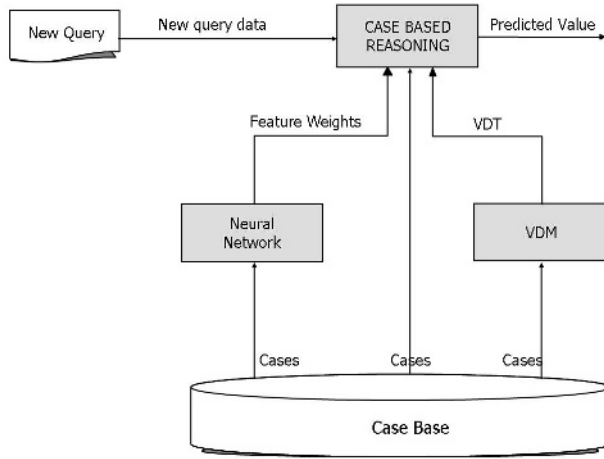
$$\delta'(x_i, y_i) = d(x_i, y_i) = \sum_{l=1}^n \left| \frac{D(f_i = x_i \cap g = c_l)}{D(f_i = x_i)} - \frac{D(f_i = y_i \cap g = c_l)}{D(f_i = y_i)} \right| \tag{8}$$

### 2.3 CANSY : A Symbolic Feature Weighting Algorithm Using a Trained Neural Network and VDM

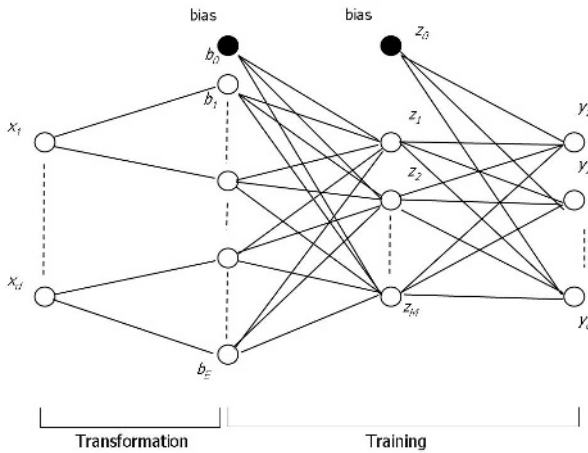
The framework of the CANSY is shown in figure 1. We extend the method for the symbolic feature case, of which all of the input features and the target features are symbolic. The learning system is consisted of three processes.

The first process is evaluating the feature weight set, which is extracted from the trained neural network. Because symbolic feature values have no distance between two values, it cannot be use as input data of a neural network.

Therefore, transformation mechanism is required for using symbolic feature values as the input data of neural network. That is, before training the neural network, all of the symbolic features are transformed into binary features like figure 2, which has  $d$  original inputs ( $x_b, i=1, \dots, d$ ),  $E$  transformed binary input nodes ( $b_b, i=1, \dots, E$ ),  $M$  hidden nodes ( $z_b, i=1, \dots, M$ ), and  $c$  output nodes ( $y_b, i=1, \dots, c$ ). Every output node represents one of possible values of the target feature, namely one of target classes. The number of output nodes is the number of possible values of the target feature.



**Fig. 1.** CBR with VDM weighting by Neural Network



**Fig. 2.** Example of a fully connected feed-forward network having transformation mechanism

When training of a neural network is finished, we obtain the feature weight set from the trained neural network using four feature weighting methods. The four feature weighting methods are Sensitivity, Activity, Saliency, and Relevance. Those methods calculate importance of each feature using the connection weights and activation patterns of nodes in the trained neural network.

1) Sensitivity: The Sensitivity of an input feature  $x_i$  is given by

$$Se_i = \frac{\left(\sum_L |P^o - P^i| / P^o\right)}{n} \tag{9}$$

where  $P^0$  is the normal prediction value for each training instance by the trained neural network and  $P^i$  is the modified prediction value when the input  $i$  is removed.  $L$  is the set of training data and  $n$  is the number of training data.

2) Activity: The activity of a hidden node  $z_j$  is given by

$$A_j = \sum_{k=1}^c w_{kj}^2 \cdot \text{var} \left( g \left( \sum_{l=0}^d w_{jl} b_l \right) \right) \tag{10}$$

where  $\text{var}()$  is the variance function. The activity of a binary input node  $b_l$  is defined as

$$A_l = \sum_{j=1}^M w_{jl}^2 \cdot A_j \tag{11}$$

and the activity of an input node  $x_i$  is given by

$$A_i = \sum_{b_l \in x_i} A_l / n_i \tag{12}$$

where  $n_i$  is the number of values of  $x_i$  input feature.

3) Saliency: The saliency of a binary input node  $b_l$  is given by

$$Sa_l = \sum_{j=1}^M \sum_{k=1}^c w_{kj}^2 \cdot w_{jl}^2 \tag{13}$$

and the saliency of an input node  $x_i$  is given by

$$Sa_i = \sum_{b_l \in x_i} Sa_l / n_i \tag{14}$$

4) Relevance: The relevance of a hidden node  $z_j$  is given by

$$R_j = \sum_{k=1}^c w_{kj}^2 \cdot \text{var}(w_{jl}) \tag{15}$$

and the overall relevance of a binary input node  $b_l$  is

$$R_l = w_{jl} \cdot R_j \tag{16}$$

and the relevance of an input node  $x_i$  is given by

$$R_i = \sum_{b_l \in x_i} R_l / n_i \tag{17}$$

The second process is constructing VDT (Value Different Tables) from the instances in the case base according to VDM, which defines the disances between different values of a given feature. For each feature, the value difference matrix is derived statistically based on the instances in the training dataset according to (18).

In fact, (18) is a simpler form of (8).

$$\delta(V_1, V_2) = \sum_{i=1}^n \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right| \tag{18}$$

Where  $V_1$  and  $V_2$  are two possible values for the feature and  $n$  is the number of classes.  $C_{1i}$  is the number of times  $V_1$  was classified into class  $i$  and  $C_i$  is the total

number of times value 1 occurred. The term  $C_i/C_j$  is the likelihood that an instance will be classified as class  $i$  given that its  $i_{th}$  feature has value  $V_j$ . Thus (18) computes overall similarity between two values by finding the sum of the difference of the likelihood over all classifications.

The third process is case-based reasoning using feature weights extracted from the trained neural network and VDT constructed by VDM. If a query is given, the distances between the query and the cases is

$$\Delta(q, x) = \sum_i^N w_i \delta(q_i, x_i)^r \quad (19)$$

where  $q$  is the query and  $x$  is a case from the case base,  $q_i$  and  $x_i$  are the  $i_{th}$  feature values of  $q$  and  $x$ , respectively. The system classifies the new instance following the closest cases from the case base.  $w_i$  is the weight of the  $i_{th}$  input feature.  $\delta(q_i, x_i)$  is the distance between two values  $q_i$  and  $x_i$  of the  $i_{th}$  input feature.  $r$  is usually set to 1 or 2 according to the case bases. In this paper, we set  $r = 1$  for all of our experiments.

### 3 Experimental Results

We apply our methods to two datasets from the UCI machine learning repository [7]. In this experiment, we created a neural network with one hidden layer. To train it, we applied the gradient descent algorithm with momentum & adaptive learning rate, which is implemented in MATLAB 6 as the default training algorithm. For calculating the weight values for input features, the four methods – Sensitivity, Activity, Saliency and Relevance are used and for calculating the distance between two symbolic features, VDT obtained from VDM is used.

We compare the performance of our methods to that of the simple k-nearest neighbor (k-nn) algorithms without feature weighting. The experiments are repeated 10 times for each dataset and in every experiment. We evaluate the performance of all methods according to k, the number of nearest neighbors which, in our experiments, takes odd numbers from 1 to 15. Table 1 shows the experimental settings for the problems.

**Table 1.** Datasets and neural network settings

Problem	Datasets				Neural Network	
	Training Instances	Test Instances	Attributes	Output Classes	# of Hidden Nodes	Training Goal
Monk's-1	124	432	6	2	2	0.01
Voting Records	300	135	16	2	4	0.01

#### 3.1 Monk's Problem

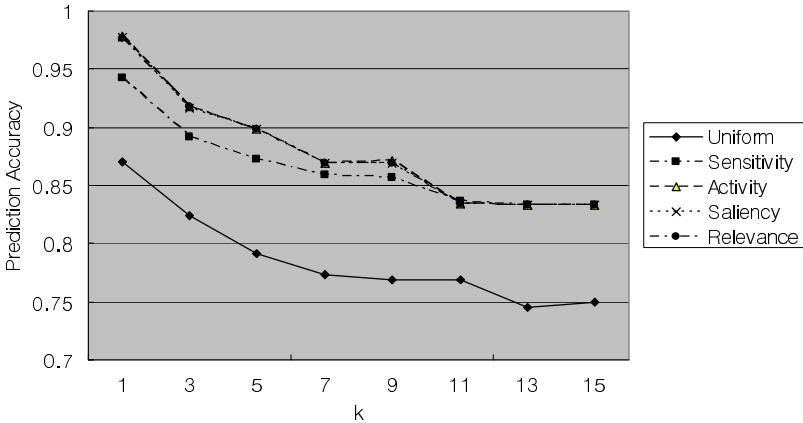
The learning task is to generalize a binary classification, robots belong either to this class or not, with a subset of all 432 possible robots give classes [9]. The domain has been partitioned into a train dataset and test dataset.

The neural network has 2 hidden neurons. We repeated the neural network training 10 times for each feature weighting method and  $k$ . Table 2 shows the experimental results.

‘Uniform’ column shows the errors of the CBR without feature weighting, that is, pure  $k$ -nn. Because we experiment with uniform method once, ‘Uniform’ column has no variance. The next four columns show the mean errors and the variances of the errors of the four method of the CBR with the proposed symbolic feature weighting.

**Table 2.** The Monk’s-1 problem – Mean errors and variances of weighting

<b>K</b>	<b>Uniform</b>	<b>Sensitivity</b>	<b>Activity</b>	<b>Saliency</b>	<b>Relevance</b>
1	0.1296	0.0576±0.0250	<b>0.0215±0.0213</b>	0.0231±0.0216	0.0234±0.0219
3	0.1759	0.1086±0.0114	0.0826±0.0130	0.0829±0.0139	<b>0.0824±0.0147</b>
5	0.2083	0.1273±0.0160	<b>0.1012±0.0081</b>	0.1014±0.0086	0.1016±0.0102
7	0.2269	0.1410±0.0061	0.1306±0.0077	0.1306±0.0075	<b>0.1303±0.0076</b>
9	0.2315	0.1435±0.0127	<b>0.1289±0.0120</b>	0.1308±0.0129	0.1306±0.0138
11	0.2315	<b>0.1630±0.0043</b>	0.1660±0.0016	0.1662±0.0015	0.1662±0.0015
13	0.2546	<b>0.1667±0.0000</b>	<b>0.1667±0.0000</b>	<b>0.1667±0.0000</b>	<b>0.1667±0.0000</b>
15	0.2500	<b>0.1667±0.0000</b>	<b>0.1667±0.0000</b>	<b>0.1667±0.0000</b>	<b>0.1667±0.0000</b>



**Fig. 3.** The Monk’s-1 : Classification accuracy of feature weighting methods

Figure 3 shows the prediction accuracy of the feature weighting methods according to  $k$ . In this problem, the proposed weighting methods show better prediction accuracy than uniform method. The difference in the prediction accuracy of the four feature weighting methods-the Sensitivity, Activity, Saliency, and Relevance, is very small and the trend in the change of the errors according to  $k$  is also similar.

### 3.2 Congressional Voting Records Database

This is the 1984 United States Congressional Voting Records Database. This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16

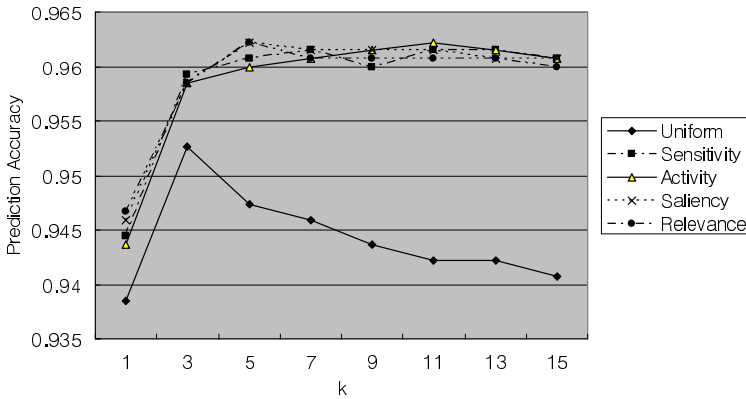
key votes. All of the instances are classified into one of 2 classes, democrat or republican.

We repeated the experiment 10 times and in every experiment and divided the 435 instances into a training dataset and a test dataset of 300 and 135, respectively. Table 3 is the experimental results. The neural network had 4 hidden neurons.

We can see that the four feature weighting methods outperform the pure  $k$ -nn in every datasets. Especially, when  $k$  is greater than 3, the difference in the prediction accuracy between the pure  $k$ -nn and proposed weighting method is very large. Figure 4 shows the prediction accuracy of the feature weighting methods according to  $k$ .

**Table 3.** The Voting Records database – Mean error of feature weighting methods

K	Uniform	Sensitivity	Activity	Saliency	Relevance
1	0.0615±0.0201	0.0556±0.0172	0.0563±0.0168	0.0541±0.0178	<b>0.0533±0.0143</b>
3	0.0474±0.0117	<b>0.0407±0.0106</b>	0.0415±0.0062	0.0415±0.0062	0.0415±0.0062
5	0.0526±0.0118	0.0393±0.0093	0.0400±0.0080	<b>0.0378±0.0107</b>	<b>0.0378±0.0107</b>
7	0.0541±0.0111	<b>0.0385±0.0104</b>	0.0393±0.0086	0.0385±0.0109	0.0393±0.0105
9	0.0563±0.0157	0.0400±0.0112	0.0385±0.0115	<b>0.0385±0.0104</b>	0.0393±0.0099
11	0.0578±0.0147	0.0385±0.0120	<b>0.0378±0.0128</b>	0.0385±0.0115	0.0393±0.0121
13	0.0578±0.0125	<b>0.0385±0.0120</b>	<b>0.0385±0.0120</b>	0.0393±0.0105	0.0393±0.0105
15	0.0593±0.0126	<b>0.0393±0.0105</b>	<b>0.0393±0.0105</b>	<b>0.0393±0.0105</b>	0.0400±0.0112



**Fig. 4.** The Voting Records : Classification accuracy of feature weighting methods

In pure  $k$ -nn, the prediction accuracy changes according to change of  $k$ . On the contrary, the four feature weighting methods prevent the decay of the prediction accuracy with the increase of  $k$ . The difference in the prediction accuracy of the four feature weighting methods is very small and the trend in the change of the errors according to  $k$  is also similar as you can see in figure 4.

## 4 Conclusion and Future Work

We adopt trained neural network for feature weighting and VDM for measuring the distance between values of symbolic features because the symbolic features need a sophisticated distance metric. The prediction accuracy of CBR, specifically the  $k$ -nn weighted by the four feature weighting methods outperforms that of pure  $k$ -nn as shown in the experiments. The four feature weighting methods adjust the effect of the input features to the distance very appropriately by weight values of features and thus enhance the prediction accuracy of the  $k$ -nn. Moreover, the feature weighting methods offer the  $k$ -nn the stability in its prediction ability. As shown in the experiments, especially in the results from the Voting Records, the accuracy doesn't decrease with the increase of  $k$  despite of decay of the prediction accuracy of the pure  $k$ -nn. In general, it is difficult to specify the optimum value for  $k$  previously. Thus that character of the feature weighting methods has significant importance because they can provide consistent accuracy to the  $k$ -nn.

The distance metrics for numeric features and symbolic features have different characteristics. We have applied the feature weighting methods by neural networks to both Euclidean distance as a representative distance metric for numeric features and VDM as a representative one for symbolic features. But many problems have mixed numeric- and nominal-valued attributes. Therefore, we have to study about many variants of VDM which have been proposed to improve heterogeneous distance functions [10].

## References

1. D. Wettschereck, D. W. Aha: Weighting features. In: Proc. ICCBR-95. A. Aamodt, M. Veloso, Eds. (1995) 347-358
2. T. Hastie, R. Tibshirani: Discriminant adaptive nearest neighbor classification. Int J IEEE Trans. Pattern Anal. Machine Intell. Vol. 18 (1996) 607-616
3. D. Wettschereck, D.W. Aha, T. Mohri: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. AI Rev. Vol. 11 (1997) 273-314
4. C.K. Shin, U.T Yun, H.K. Kim, S.C. Park: A Hybrid Approach of Neural Network and Memory-Based Learning to Data Mining. Int J IEEE Trans. on Neural Networks. Vol. 11, No. 3 (2000) 637-646
5. Stanfill, C., Waltz, D.: Toward memory-based reasoning. Communications of the ACM. Vol. 29, No. 12 (1996) 1213-1228
6. Scott Cost, Steven Salzberg: A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine Learning. Vol. 10 (1993) 57-78
7. C. Blake, E. Keogh, C. J. Merz: UCI Repository of Machine Learning Databases. In: Univ. California, Dept. Inform. Comput. Sci., Irvine, CA. (1999)
8. B. Shneiderman: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Reading, MA: Addison-Wesley (1997)
9. S.B. Thrun et al.: A Performance Comparison of Different Learning Algorithms. In: Carnegie Mellon University (1991)
10. D. Randall Wilson, Tony R. Martinez: Improved Heterogeneous Distance Functions. J Artificial Intelligence Research, Vol. 6 (1997) 1-34



# Spatio-temporal Rule Mining: Issues and Techniques

Győző Gidófalvi<sup>1</sup> and Torben Bach Pedersen<sup>2</sup>

<sup>1</sup> Geomatic aps — center for geoinformatik  
gyg@geomatic.dk

<sup>2</sup> Aalborg University — Department of Computer Science  
tbp@cs.aau.dk

**Abstract.** Recent advances in communication and information technology, such as the increasing accuracy of GPS technology and the miniaturization of wireless communication devices pave the road for Location-Based Services (LBS). To achieve high quality for such services, spatio-temporal data mining techniques are needed. In this paper, we describe experiences with spatio-temporal rule mining in a Danish data mining company. First, a number of real world spatio-temporal data sets are described, leading to a taxonomy of spatio-temporal data. Second, the paper describes a general methodology that transforms the spatio-temporal rule mining task to the traditional market basket analysis task and applies it to the described data sets, enabling traditional association rule mining methods to discover spatio-temporal rules for LBS. Finally, unique issues in spatio-temporal rule mining are identified and discussed.

## 1 Introduction

Several trends in hardware technologies such as display devices and wireless communication combine to enable the deployment of mobile, Location-based Services (LBS). Perhaps most importantly, global positioning systems (GPS) are becoming increasingly available and accurate. In the coming years, we will witness very large quantities of wirelessly Internet-worked objects that are location-enabled and capable of movement to varying degrees. These objects include consumers using GPRS and GPS enabled mobile-phone terminals and personal digital assistants, tourists carrying on-line and position-aware cameras and wrist watches, vehicles with computing and navigation equipment, etc.

These developments pave the way to a range of qualitatively new types of Internet-based services [8]. These types of services, which either make little sense or are of limited interest in the context of fixed-location, desktop computing, include: traffic coordination and management, way-finding, location-aware advertising, integrated information services, e.g., tourist services.

A single generic scenario may be envisioned for these location-based services. Moving service users disclose their positional information to services, which use this and other information to provide specific functionality. To customize the

interactions between the services and users, data mining techniques can be applied to discover interesting knowledge about the behaviour of users. For example, groups of users can be identified exhibiting similar behaviour. These groups can be characterized based on various attributes of the group members or the requested services. Sequences of service requests can also be analyzed to discover regularities in such sequences. Later these regularities can be exploited to make intelligent predictions about user's future behaviour given the requests the user made in the past. In addition, this knowledge can also be used for delayed modification of the services, and for longer-term strategic decision making [9].

An intuitively easy to understand representation of this knowledge is in terms of rules. A *rule* is an implication of the form  $A \Rightarrow B$ , where  $A$  and  $B$  are sets of attributes. The idea of mining association rules and the subproblem of mining frequent itemset was introduced by Agrawal et al. for the analysis of market basket data [1]. Informally, the task of mining frequent itemsets can be defined as finding all sets of items that co-occur in user purchases more than a user-defined number of times. The number of times items in an itemset co-occur in user purchases is defined to be the *support* of the itemset. Once the set of high-support, so called *frequent* itemsets have been identified, the task of mining association rules can be defined as finding disjoint subsets  $A$  and  $B$  of each frequent itemset such that the conditional probability of items in  $B$  given the items in  $A$  is higher than a user-defined threshold. The conditional probability of  $B$  given  $A$  is referred to as the *confidence* of the rule  $A \Rightarrow B$ . Given that coffee and cream are frequently purchased together, a high-confidence rule might be that "60% of the people who buy coffee also buy cream." Association rule mining is an active research area. For a detailed review the reader is referred to [5].

Spatio-temporal (ST) rules can be either *explicit* or *implicit*. Explicit ST rules have a pronounced ST component. Implicit ST rules encode dependencies between entities that are defined by spatial (north-of, within, close-to, . . .) and/or temporal (after, before, during, . . .) predicates. An example of an explicit ST rule is: "Businessmen drink coffee at noon in the pedestrian street district." An example of an implicit ST rule is: "Middle-aged single men often co-occur in space and time with younger women." In this paper, we describe our experiences with ST rule mining in the Danish spatial data mining company, Geomatic.

The task of finding ST rules is challenging because of the high cardinality of the two added dimensions: space and time. Additionally, straight-forward application of association rule mining methods cannot always extract all the interesting knowledge in ST data. For example, consider the previous implicit ST rule example, which extracts knowledge about entities (people) with different attributes (gender, age) that interact in space and time. Such interaction will not be detected when association rule mining is applied in straight-forward manner. This creates a need to explore the special properties of ST data in relation to rule mining, which is the focus of this paper.

The contributions of the paper are as follows. First, a number of real world ST data sets are described, and a taxonomy for ST data is derived. Second, having the taxonomy, the described data sets, and and the desirable LBSes

in mind, a general methodology is devised that projects the ST rule mining task to traditional market basket analysis. The proposed method can in many cases efficiently eliminate the above mentioned explosion of the search space, and allows for the discovery of both implicit and explicit ST rules. Third, the projection method is applied to a number of different type of ST data such that traditional association rule mining methods are able to find ST rules which are useful for LBSes. Fourth, as a natural extension to the proposed method, spatio-temporally restricted mining is described, which in some cases allows for further quantitative and qualitative mining improvements. Finally, a number of issues in ST rule mining are identified, which point to possible future research directions.

Despite the abundance of ST data, the number of algorithms that mine such data is small. Since the pioneering work of [2], association rule mining methods were extended to the spatial [3,4,6,11], and later to the temporal dimension [12]. Other than in [13,15], there has been no attempts to handle the combination of the two dimensions. In [15] an efficient depth-first search style algorithm is given to discover ST sequential patterns in weather data. The method does not fully explore the spatial dimension as no spatial component is present in the rules, and no general spatial predicate defines the dependencies between the entities. In [13], a bottom-up, level-wise, and a faster top-down mining algorithm is presented to discover ST periodic patterns in ST trajectories. While the technique can naturally be applied to discover ST event sequences, the patterns found are only within a single event sequence.

The remainder of the paper is organized as follows. Section 2 introduces a number of real world ST data sets, along with a taxonomy of ST data. In Section 3, a general methodology is introduced that projects the ST rule mining task to the traditional market basket analysis or frequent itemset mining task. The proposed problem projection method is also applied to the example data sets such that traditional association rule mining methods are able to discover ST rules for LBSes. Finally, Sections 4 and 5 identify unique issues in ST rule mining, conclude, and point to future work.

## 2 Spatio-temporal Data

Data is obtained by measuring some attributes of an entity/phenomena. When these attributes depend on the place and time the measurements are taken, we refer to it as ST data. Hence such ST measurements not only include the measured attribute values about the entity or phenomena, but also two special attribute values: a location value, *where* the measurement was taken, and a time value, *when* the measurement was taken. Disregarding these attributes, the non-ST rule “Businessmen drink coffee” would result in annoying advertisements sent to businessmen who are in the middle of an important meeting.

**Examples of ST Data Sets.** The first ST data set comes from the “Space, Time, and Man” (STM) project [14]—a multi-disciplinary project at Aalborg University. In the STM project activities of thousands of individuals are continuously registered through GPS-enabled mobile phones, referred to as mobile

terminals. These mobile terminals, integrated with various GIS services, are used to determine close-by services such as shops. Based on this information in certain time intervals the individual is prompted to select from the set of available services, which s/he currently might be using. Upon this selection, answers to subsequent questions can provide a more detailed information about the nature of the used service. Some of the attributes collected include: location and time attributes, demographic user attributes, and attributes about the services used. This data set will be referred to as STM in the following.

The second ST data set is a result of a project carried out by the Greater Copenhagen Development Council (Hovedstadens Udviklings Råd (HUR)). The HUR project involves a number of city busses each equipped with a GPS receiver, a laptop, and infrared sensors for counting the passengers getting on and off at each bus stop. While the busses are running, their GPS positions are continuously sampled to obtain detailed location information. The next big project of HUR will be to employ chip cards as payment for the travel. Each passenger must have an individual chip card that is read when getting on and off the bus. In this way an individual payment dependent on the person and the length of the travel can be obtained. The data recorded from the chip cards can provide valuable passenger information. When analyzed, the data can reveal general travel patterns that can be used for suggesting new and better bus routes. The chip cards also reveal individual travel patterns which can be used to provide a customized LBS that suggests which bus to take, taking capacities and correct delays into account. In the following, the datasets from the first and second projects of HUR will be referred to as HUR1 and HUR2, respectively.

The third ST data set is the publicly available INFATI data set [7], which comes from the intelligent speed adaptation (INtelligent FArTIlpasning (INFATI)) project conducted by the Traffic Research Group at Aalborg University. This data set records cars moving around in the road network of Aalborg, Denmark over a period of several months. During this period, periodically the location and speeds of the cars are sampled and matched to corresponding speed limits. This data set is interesting, as it captures the movement of private cars on a day-to-day basis, i.e., the daily activity patterns of the drivers. Additional information about the project can be found in [10]. This data set will be referred to as INFATI in the following.

Finally, the last example data set comes from the Danish Meteorology Institute (DMI) and records at fixed time intervals atmospheric measurements like temperature, humidity, and pressure for Denmark for 5 km grid cells. This data set is unique in that unlike the other datasets it does not capture ST characteristics of moving objects, but nonetheless is ST. This data set will be referred to as DMI in the following.

**A Taxonomy of ST Data.** Data mining in the ST domain is yet largely unexplored. There does not even exist any generally accepted taxonomy of ST data. To analyze such data it is important to establish a taxonomy.

Perhaps the most important criterion for this categorization is whether the measured entities are *mobile* or *immobile*. The ST data in the DMI data set is

immobile in the sense that the temperature or the amount of sunshine does not move from one location to the other, but rather, as a continuous phenomenon, changes its attribute value over time at a given location. On the other hand, the observed entities in the other four datasets are rather mobile.

Another important criterion for categorization is whether the attribute values of the measured entities are *static* or *dynamic*. There are many examples of static attributes values but perhaps one that all entities possess is a unique identifier. Dynamic attributes values change over time. This change can be slow and gradual, like in the case of the age of an observed entity, or swift and abrupt, like in the case of an activity performed by the observed entity, which starts at a particular time and last for a well-specified time interval only.

### 3 Spatio-temporal Baskets

Following the methodology of market basket analysis, to extract ST rules for a given data set, one needs to define ST *items* and *baskets*. This task is important, since any possible knowledge that one can extract using association rule mining methods will be about the possible dependencies of the items within the baskets.

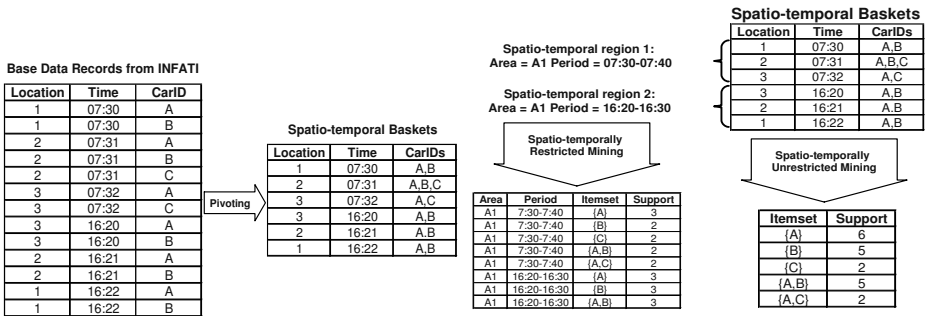
**Mobile Entities with Static and Dynamic Attributes.** Consider the STM data; it is mobile in nature and has several static and dynamic attributes. Base data contains the identity and some demographic attributes of the user, and the activity performed by user at a particular location and time. Further attributes of the locations where the activity is performed are also available. By applying association rule mining on this base data one can find possible dependencies between the activities of the users, the demographics of the users, the characteristics of the locations there the activities are performed, and the location and time of the activities. Since the location and time attributes are items in the baskets one may find {Strøget,noon,businessman,café} as a frequent itemset and from it the association rule {Strøget,noon,businessman}  $\Rightarrow$  {café}. Strøget being a famous pedestrian street district in central Copenhagen in Denmark, this rule clearly has both a spatial and temporal component and can be used to advertise special deals of a café shop on Strøget to businessmen who are in the area around noon.

In the INFATI data set, a record in the base data contains a location, a time, a driver identifier, and the current speed of the car along with the maximum allowed speed at the particular location. The possible knowledge one can discover by applying association rule mining on the base data is where and when drivers or a particular driver occur(s) and/or speed(s) frequently. However, one may in a sense pivot this table of base data records such that each new row represents an ST region and records the car identifiers that happen to be in that region. Applying association rule mining on these ST baskets one may find which cars co-occur frequently in space and time. Such knowledge can be used to aid intelligent rideshare services. It can also be valuable information for constructing traffic flow models and for discovering travel patterns. While the possible knowledge discovered may be valuable for certain applications, the extracted rules are

not clearly ST, i.e.: there is no *explicit* ST component in them. In fact the same set of cars may frequently co-occur at several ST regions which may be scattered in space and time. Nonetheless, it can be argued that since the “co-occurrence” between the items in the ST baskets is actually an ST predicate in itself, the extracted rules are *implicitly* ST.

An alternative to this approach might be to restrict the mining of the ST baskets to larger ST regions. While this may seem useless at first, since the baskets themselves already define more fine-grained ST regions, it has several advantages. First, it allows the attachment of an explicit ST component to each extracted rule. Second, it enhances the quality of the extracted rules. Finally, it significantly speeds up the mining process, as no two itemsets from different regions are combined and tried as a candidate. Figure 1 shows the process of pivoting of some example records abstracted from the INFATI data set. Figure 2 shows the process and results of spatio-temporally restricted and unrestricted mining of the ST baskets. In this example the shown frequent itemsets are based on an absolute minimum support of 2 in both cases, however in the restricted case specifying a relative minimum support would yield more meaningful results. Naturally the adjective “relative” refers to the number of baskets in each of the ST regions. Figure 2 also shows the above mentioned qualitative differences in the result obtained from spatio-temporally restricted vs. unrestricted mining. While the frequent co-occurrence of cars A and B, and cars A and C are detected by unrestricted mining, the information that cars A and B are approximately equally likely to co-occur in area A1 in the morning as in the afternoon, and that cars A and C only co-occur in area A1 in the morning is missed.

Similar pivoting techniques based on other attributes can also reveal interesting information. Consider the data set in HUR2 and the task of finding frequently traveled routes originating from a given ST region. In the HUR2 data set a record is generated every time a user starts and finishes using a transportation service. This record contains the identifier of the user, the transportation line used, and the location and time of the usage. For simplicity assume that a trip is defined to last at most 2 hours. As a first step of the mining, one can retrieve all the



**Fig. 1.** Process of pivoting to obtain ST baskets from INFATI base data

**Fig. 2.** Process and results of spatio-temporally restricted vs. unrestricted mining of ST baskets

records that fall within the ST region of the origin. Following, one can retrieve all the records within 2 hours of the users that belonged to the first set. By pivoting on the user-identifiers, one can derive ST baskets that contain locations where the user generated a record by making use of a transportation service. Applying association rule mining to the so-derived ST baskets one may find frequently traveled routes originating from a specific ST region. The pivoting process for obtaining such ST baskets and the results of mining such baskets is illustrated in a simple example in the light bordered box of Figure 3. Naturally, the frequent itemset mining is only applied to the "Unique Locations" column of the ST baskets. As before the minimum support is set to 2. Considering the spatial relation between the locations one might consider altering the bus routes to better meet customer needs. For example, if locations A and C are close by on the road network, but no bus line exists with a suitable schedule between A and C, then in light of the evidence, i.e., support of A,B,C is 2, such a line can be added. Note that while the discovered frequent location sets do not encode any temporal relation between the locations, one can achieve this by simply placing ST regions into the ST baskets as items. The pivoting process and the results of mining are shown in the dark bordered box of Figure 3. The discovered ST itemsets can help in adjusting timetables of busses to best meet customer needs.

**Immobile Entities with Static and Dynamic Attributes.** So far the examples considered datasets that are mobile and have either static, dynamic, or both types of attribute values, as the DMI data set. The base data can be viewed as transactions in a relational table with a timestamp, a location identifier and some atmospheric measurements like temperature, humidity, and pressure. Considering the geographical locations A, B, C, and D depicted in Figure 4, we might be interested in trends like, when the temperature in regions A and B is high and the pressure in regions A and C is low, then at the same time the humidity in region D is medium. By applying something similar to the pivoting techniques above, we can extract such information as follows. For each record concatenate the location identifiers with the atmospheric measurements. Then, for each dis-

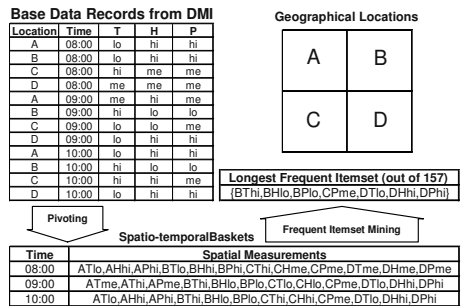
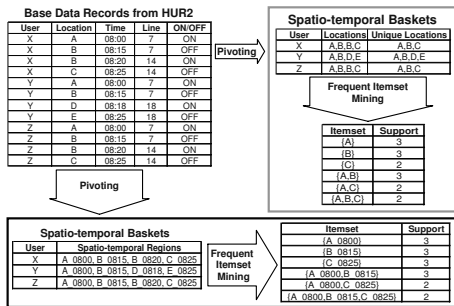


Fig. 3. ST baskets and frequent itemset mining for HUR2

Fig. 4. ST baskets and frequent itemset mining of DMI

tinct time interval when measurements are taken, put all concatenated values, each of which is composed of a location identifier and an atmospheric measurement, into a single, long ST basket. By performing association mining on the derived ST baskets one can obtain the desired knowledge.

As an illustrative example, depicted in Figure 4, consider the four neighboring cells A, B, C, and D and the corresponding measurements of temperature (T), humidity (H), and pressure (P) at three different times. Items in the ST baskets are derived by concatenating a location identifier followed by an attribute symbol and an attribute value. Hence, the item ‘ATlo’ in the ST basket at time ‘08:00’ encodes the fact that at ‘08:00’ at location ‘A’ the temperature (‘T’) was low (‘lo’). Notice that the extracted knowledge refers to specific locations. If one is interested in obtaining knowledge about the inter-dependencies of these attributes relative (in space) to one another, for each base data record at each distinct time interval when measurements are taken, an ST basket can be constructed that encodes measurements from neighboring cells only. So, for example considering the immediate 8 neighbors of a cell and assuming three different attributes the number of items in each basket is  $3 + 8 \times 3 = 27$ . Considering a five-by-five relative neighborhood centered around a cell the number of items in each basket is 75, and the number of possible itemsets, given three possible attribute values for each of the attributes is  $3^{75} \approx 6.1 \times 10^{34}$ . To reduce complexity, top-down and bottom-up mining can occur at different spatial and temporal granularities.

While in the above examples the type of ST data that was analyzed and the type of ST knowledge that was extracted is quite different the underlying problem transformation method—referred to as *pivoting*—is the same. In general, one is given base records with two sets of attributes  $A$  and  $B$ , which are selected by a data mining expert and can contain either spatial, temporal and/or ordinary attributes. Pivoting is then performed by grouping all the base records based on the  $A$ -attribute values and assigning the  $B$ -attribute values of base records in the same group to a single basket. Bellow, attributes in  $A$  are referred to as *pivoting* attributes or *predicates*, and attributes in  $B$  are referred to as *pivoted* attributes or *items*. Depending on the type of the pivoting attributes and the type of the pivoted attributes the obtained baskets can be either *ordinary*, *spatial*, *temporal*, or *ST* baskets. Table 1 shows the different types of baskets as a function of the different types of predicates used to construct the baskets and the different types of items placed in the baskets. The symbols s, t, st, i, and b in the table are used to abbreviate the terms ‘spatial’, ‘temporal’, ‘spatio-temporal’, ‘items’, and ‘baskets’ respectively.

In the “co-occurrence” mining task, which was earlier illustrated on the IN-FATI data, the concept of restricted mining is introduced. This restriction is possible due to a side effect of the pivoting technique. When a particular basket is constructed, the basket is assigned the value of the pivoting attribute as an implicit label. When this implicit basket label contains a spatial, temporal, or ST component, restricting the mining to a particular spatial, temporal, or ST subregion becomes a natural possibility. It is clear that not all basket types can



**Table 1.** Types of baskets as a function of predicate type and item type **Table 2.** Possible mining types of different types of baskets

pred/item type	s-i	t-i	st-i	ordinary-i	basket/mining type	s-r	t-r	st-r	unr
s-predicate	s-b	st-b		s-b	s-basket	X			X
t-predicate	st-b	t-b		t-b	t-basket		X		X
st-predicate	st-b	st-b	st-b	st-b	st-basket	X	X	X	X
other-predicate	s-b	t-b	st-b	ordinary-b	other-basket				X

be mined using spatial, temporal, or ST restrictions. Table 2 shows for each basket type the type of restrictions for mining that are possible. The symbols s, t, st, r, and unr in the table are used to abbreviate the terms ‘spatial’, ‘temporal’, ‘spatio-temporal’, ‘restricted’, and ‘unrestricted’ respectively.

### 4 Issues in Spatio-temporal Rule Mining

The proposed pivoting method naturally brings up questions about feasibility and efficiency. In cases where the pivoted attributes include spatial and/or temporal components, the number of items in the baskets is expected to be large. Thus, the number and length of frequent itemsets or rules is expected to grow. Bottom-up, level-wise algorithms are expected to suffer from excessive candidate generation, thus top-down mining methods seem more feasible. Furthermore, due to the presence of very long patterns, the extraction of all frequent patterns has limited use for analysis. In such cases closed or maximal frequent itemsets can be mined.

Useful patterns for LBSEs are expected to be present only in ST subregions, hence spatio-temporally restricted rule mining will not only make the proposed method computationally more feasible, but will also increase the quality of the result. Finding and merging patterns in close-by ST subregions is also expected to improve efficiency of the proposed method and the quality of results.

Placing concatenated location and time attribute values about individual entities as items into an ST basket allows traditional association rule mining methods to extract ST rules that represent ST event sequences. ST event sequences can have numerous applications, for example an intelligent ridesharing application, which finds common routes for a set of commuters and suggests rideshare possibilities to them. Such an application poses a new requirement on the discovered itemsets, namely, they primarily need to be “long” rather than frequent (only a few people will share a given ride, but preferably for a long distance). This has the following implications and consequences. First, all subsets of frequent and long itemsets are also frequent, but not necessarily long and of interest. Second, due to the low support requirement a traditional association rule mining algorithm, disregarding the length requirement, would explore an excessive number of itemsets, which are frequent but can never be part of a long and frequent itemset. Hence, simply filtering out “short” itemsets after the mining process is inefficient and infeasible. New mining methods are needed that efficiently use the length criterion during the mining process.

## 5 Conclusion and Future Work

Motivated by the need for ST rule mining methods, this paper established a taxonomy for ST data. A general problem transformation method was introduced, called pivoting, which when applied to ST datasets allows traditional association rule mining methods to discover ST rules. Pivoting was applied to a number of ST datasets allowing the extraction of both explicit and implicit ST rules useful for LBSes. Finally, some unique issues in ST rule mining were identified, pointing out possible research directions.

In future work, we will devise and empirically evaluate algorithms for both general and spatio-temporally restricted mining, and more specialized types of mining such as the ridesharing suggestions. Especially, algorithms that take advantage of the above-mentioned “long rather than frequent” property of rideshare rules will be interesting to explore.

## References

1. R. Agrawal, T. Imilienski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *Proc. of SIGMOD*, pp. 207–216, 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of VLDB*, pp. 487–499, 1994.
3. M. Ester, H.-P. Kriegel, and J. Sander. Spatial Data Mining: A Database Approach. In *Proc. of SSD*, pp. 47–66, 1997.
4. M. Ester, A. Frommelt, H.-P. Kriegel, and J. Sander. Algorithms for Characterization and Trend Detection in Spatial Databases. In *Proc. of KDD*, pp. 44–50, 1998.
5. B. Goethals. Survey on frequent pattern mining. Online at: [citeseer.ist.psu.edu/goethals03survey.html](http://citeseer.ist.psu.edu/goethals03survey.html)
6. J. Han, K. Koperski, N. Stefanovic. GeoMiner: A System Prototype for Spatial Data Mining. In *Proc. of SIGMOD*, pp. 553–556, 1997.
7. INFATI. The INFATI Project Web Site: [www.infati.dk/uk](http://www.infati.dk/uk)
8. C. S. Jensen. Research Challenges in Location-Enabled M-Services. In *Proc. of MDM*, pp. 3–7, 2003.
9. C. S. Jensen, A. Kligys, T. B. Pedersen, and I. Timko. Multidimensional Data Modeling for Location-Based Services. *VLDB Journal*, 13(1):1–21, 2004.
10. C. S. Jensen, H. Lahrmann, S. Pakalnis, and S. Runge. (2004) The INFATI data. Time Center TR-79, [www.cs.aau.dk/TimeCenter](http://www.cs.aau.dk/TimeCenter).
11. K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Proc. of SSD*, pp. 47–66, 1995.
12. Y. Li, X. S. Wang, and S. Jajodia. Discovering Temporal Patterns in Multiple Granularities. In *Proc. of TSDM*, pp. 5–19, 2000.
13. N. Mamoulis, H. Cao, G. Kollis, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, Indexing, and Querying Historical Spatiotemporal Data. In *Proc. of KDD*, pp. 236–245, 2004.
14. STM. Space, Time Man Project Web Site: [www.plan.aau.dk/~hhh/](http://www.plan.aau.dk/~hhh/)
15. I. Tsoukatos and D. Gunopulos. Efficient Mining of Spatiotemporal Patterns. In *Proc. of SSTD*, pp. 425–442, 2001.

# Hybrid Approach to Web Content Outlier Mining Without Query Vector

Malik Agyemang<sup>1</sup>, Ken Barker<sup>1</sup>, and Reda Alhajj<sup>1,2</sup>

<sup>1</sup>Department of Computer Science, University of Calgary, Calgary, Alberta, Canada

<sup>2</sup>Department of Computer Science, Global University, Beirut, Lebanon  
{agyemang, barker, alhajj}@cpsc.ucalgary.ca

**Abstract.** Mining outliers from large datasets is like finding needles in a haystack. Even more challenging is sifting through the dynamic, unstructured, and ever-growing web data for *outliers*. This paper presents *HyCOQ*, which is a hybrid algorithm that draws from the power of n-gram-based and word-based systems. Experimental results obtained using embedded motifs without a dictionary show significant improvement over using a domain dictionary irrespective of the type of data used (words, n-grams, or hybrid). Also, there is remarkable improvement in recall with hybrid documents compared to using raw words and n-grams without a domain dictionary.

## 1 Introduction

Web content mining, deals with classifying web pages into categories based on their content so that similar pages can be grouped together to enhance performance. The varying content of the web makes web content mining very challenging and wide ranging encompassing several fields in computer science (e.g., multimedia, IR, data mining, artificial intelligence, etc.).

Many intelligent systems have been developed for mining web information. However, only a few [1, 2] are dedicated to finding web documents with *varying content* from their parent category, called *web content outliers*. Web content outliers are web documents with different content compared to similar documents taken from the same category [1]. Mining web content outliers may lead to the discovery of emerging business trends, improvement in the quality of results returned from search engines, and the identification of competitors in e-commerce [1, 2], among others.

This paper proposes the *HyCOQ* algorithm for mining web content outliers which uses hybrid data without a dictionary. *HyCOQ* uses IR techniques to extract useful features from the documents and then applies dissimilarity algorithms to determine outlying documents based on computed *nearest dissimilarity densities*.

The *HyCOQ* algorithm which uses hybrid data as input without using a query vector (domain dictionary). The hybrid data has the added advantage of the strengths of both word-based and n-grams-based systems. The experimental results using embedded motifs without a domain dictionary produces an improvement of more than 100% in recall over using a domain dictionary. Similarly, using hybrid documents show

remarkable improvements in recall irrespective of whether a domain dictionary is used or not. A very importance feature of HyCOQ is that it is domain independent. Motifs are defined as documents or sets of documents taken from a known category other than the category being mined. They are referred to as embedded motifs if they are combined with the category being mined.

The rest of the paper is organized as follows. Section 2 presents a detailed review of outlier mining techniques. Section 3 discusses the hybrid web content outlier mining approach and the HyCOQ algorithm. The experimental results are reported in Section 4. Conclusions and future work are in Section 5.

## 2 Related Work

In statistics, outliers are data objects that show different characteristics from standard distributions. Over a hundred tests, called discordancy tests, have been developed for different scenarios [4]. The statistical techniques require *a priori* knowledge of the data distribution (e.g., Normal, Poisson) and distribution parameters (e.g., mean, variance), which is a major setback. In addition, most of the distributions used are univariate. Depth-based techniques represent every data object in a  $k$ - $d$  space and assign a depth to each object [12]. Data objects with smaller depths are declared outliers. Depth-based algorithms become inefficient for higher dimensional data ( $k \geq 4$ ) because they rely on the computation of  $k$ - $d$  convex hull, which has lower bound complexity of  $\Omega(n^{k/2})$  for  $n$  objects, where  $k$  is the number of dimensions.

The distance-based technique assigns numeric distances to data objects and computes outliers as data objects with relatively larger distances [13, 14]. It generalizes many of the existing notions of outliers, and enjoys better computational complexity than depth-based techniques for higher dimensional data. Ramaswamy *et al* [16] use distance to the  $k$ -nearest neighbor to rank outliers. Efficient algorithm for computing the top- $n$  outliers using their rankings is provided. However, distance-based outlier algorithms are not capable of detecting all forms of outliers [3]. The local outlier factor technique addresses this major setback. It assigns a '*degree of outlying*' to every object and declares data objects with high local outlier factor values as outliers [3, 11]. The local outlier factor depends on the remoteness of an object from its immediate neighbors. Every object is treated as potential outlier; this can capture all the different forms of outliers that are ignored by the earlier algorithms.

The algorithms discussed so far are designed solely for numeric data. To find non numeric outliers, Liu *et al* [15] propose algorithms for finding unexpected information from a competitor's website. Though very successful on real world datasets, there are major differences between their approach and our work. Their approach requires a query vector. It also requires the miner to identify a competitor's website before the mining process begins. Their approach finds *interesting and unexpected patterns*, which may not necessarily be outlying patterns. Agyemang *et al* [1, 2] propose the general framework for mining web outliers. Algorithms based on  $n$ -grams and words are proposed for mining web content outliers. The major drawback is that both algorithms use domain dictionary. In addition  $n$ -grams tend to be slow for large datasets and words do not support partial matching of strings.

### 3 A Hybrid Web Content Outlier Mining Approach

N-grams are  $n$ -continuous character slices of a larger string into smaller strings each of size  $n$ . In general, the string can be a single word or a set of words occurring together in a document. This paper uses  $n$ -grams as an  $n$ -consecutive character slice of a word into smaller substrings each of size  $n$ . N-gram techniques have been applied in different domains such as language identification, document categorization, robust handling of noisy text, and many other domains of natural language processing [5, 9]. The success of  $n$ -gram-based systems arises because strings are decomposed into smaller parts causing errors to affect only a limited number of parts rather than the whole string. The number of higher order  $n$ -grams common to two strings is a measure of their similarity, which is resistant to a large variety of textual errors [9].

In word-based systems, direct comparison of words to the dictionary is performed. Stemming algorithms are usually applied to convert similar but different words to their root terms. For example, *educate*, *educational*, and *educated* are each replaced with *education*. Similarity algorithms declare two documents as similar if a reasonable number of words from the document and the query vector match in precision and recall. Word-based and  $n$ -gram-based systems have been thoroughly studied in IR for document classification. The strengths of word-based systems translate to weaknesses of  $n$ -gram-based systems and vice versa. For example,  $n$ -gram-based systems are very efficient in memory utilization because they have fixed lengths compared to words with variable lengths. However,  $n$ -gram-based systems become slow for very large datasets because of the huge number of  $n$ -gram vectors generated during mining. Similarly,  $n$ -gram-based systems support partial matching of strings, which is a good property for outlier detection whereas word-based systems do not.

#### 3.1 The HyCOQ Algorithm

The algorithm draws from the strengths of  $n$ -gram-based and word-based systems, while eliminating the weaknesses in both systems. It also takes advantage of the html structure of the web by assigning weights to html tags containing the keywords in the documents. The texts are extracted from the documents and processed. The extracted keywords are merged with higher order  $n$ -grams to form the hybrid documents. Hybrid document profiles are generated and used as input to the outlier detection algorithm. The  $k$ -dissimilarity algorithm uses pair-wise document dissimilarities as input to generate  $k$ -dissimilarities, *neighborhood dissimilarities* and *nearest dissimilarity densities* for each document. Documents with high nearest dissimilarity densities are more likely to be outliers than those with very low densities.

The HyCOQ algorithm shown in Figure 1 does not use a domain dictionary unlike its predecessors WCO-Mine [1] and WCON-Mine [2] for the following reasons. First, using a dictionary (query vector) requires the services of a domain expert, which can be very expensive and error prone. Second, using a dictionary can produce unrealistic and global outliers. The documents in the category do not have much influence on the type of outliers produced. Thus, the dictionary becomes more important than the documents themselves. Third, the mining process can produce false positive results if the dictionary is over exhaustive (i.e., the dictionary covers more than the domain being mined) and false negative results otherwise. Finally, using a dictionary means a domain must be identified *a priori* for the mining process to begin. Thus, data with unidentified domains cannot be mined.

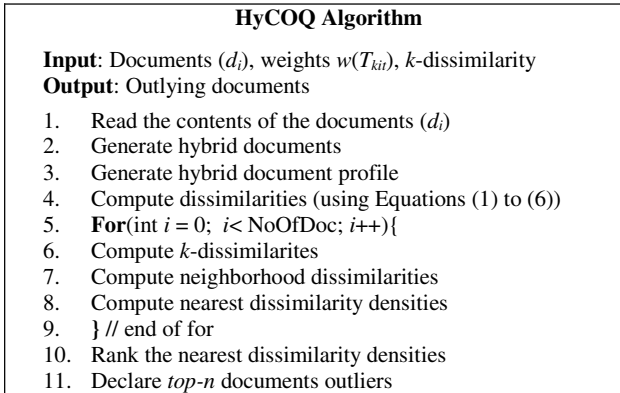


Fig. 1. The HyCOQ Algorithm

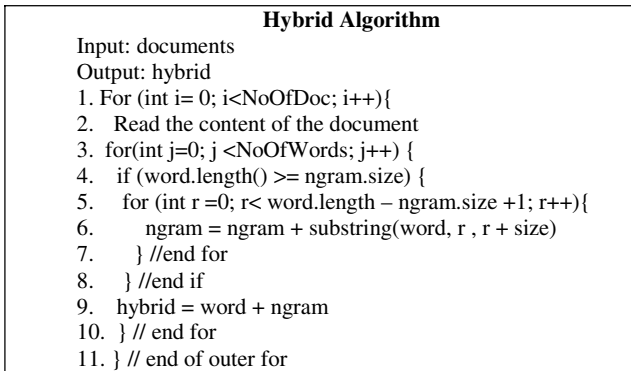


Fig. 2. The Hybrid Algorithm

### 3.1.1 Extraction and Preprocessing

The extraction involves downloading the required documents from the web and extracting the text contained in the html tags (i.e., <Title>, <Meta> and <Body>) for further processing. The paper uses already extracted dataset taken from the *webkb* data repository [18]. During preprocessing, any data besides text embedded in the html tags are removed. Symbols, numeric characters, and stop-words are also removed. The stop-word removal is done using independent publicly available list of stop-words [19]. Keywords are grouped under their respective html tags.

### 3.1.2 N-Gram Generation and Hybridization

This phase generates higher order  $n$ -grams for each document. The  $n$ -grams of a string of length  $k$  are obtained by sliding a window of size  $n$  over the string and recording the characters that appear in the window at every position. For example, the string *student* has ‘*stude*’, ‘*tuden*’, ‘*udent*’ 5-grams. The maximum number of  $n$ -grams generated from a string of length  $k$  is  $(k-n+1)$ . The hybrid documents are

formed by merging the n-grams with the keywords. The hybridization algorithm is shown in Figure 2.

**3.1.3 Hybrid Document Profile Generation**

The hybrid documents are compared, and the number of times a term appears in each document is identified. The results are hashed to keep track of terms and their frequency in each document. The terms and their respective counts are sorted and output to a file. The sorted files constitute the hybrid document profiles which are used as input to the outlier detection algorithm.

**3.1.4 Outlier Detection**

The goal of HyCOQ is to compute dissimilarity for determining documents with varying content. The paper employs the vector space model as a representation scheme. Each document is represented as a vector of indexed terms (keywords). Weights are assigned to terms within a document based on html tags that enclosed their root words and whether or not a term occurs in a document. Let  $t_i$  be an indexed term and  $z$  be the number of terms in all documents so  $T = \{t_1, t_2, t_3, \dots, t_z\}$  is the set of all terms. A document  $D_i$  characterized by a set of index terms  $T_k$  is represented in vector space model as follows:

$$D_i = (W_{i1}, W_{i2}, W_{i3}, \dots, W_{ik}) \tag{1}$$

where  $W_{ij}$  represents the weight of term  $T_j$  in document  $D_i$ ;  $W_{ij} > 0$  if term  $T_j$  appears in a document and zero otherwise. The weight  $W_{ij}$  is obtained by adopting Salton’s term-weighting formula [17].

$$w_{ik} = \frac{tf_{ik} \cdot \log \left( \frac{N}{n_k} \right)}{\sqrt{\sum_i (tf_{ik})^2 \cdot \left( \log \left( \frac{N}{n_k} \right) \right)^2}} \tag{2}$$

where  $tf_{ik}$  is the term frequency of document  $D_i$ ,  $N$  is the size of the document collection and  $n_k$  represents the number of documents with term  $T_k$ . The denominator in Equation (2) is used for content normalization which ensures relevant documents are ranked accordingly. The algorithm assigns specific importance to the html tags that contain the terms. In particular, terms contained in metadata (i.e., <meta>, <title>) are assigned larger weights than terms contained in the body tag using the function:

$$w(T_{kit}) = \begin{cases} 1 & \text{if } T_k \in \text{metadata tag} \\ \lambda & \text{otherwise, } 0 < \lambda < 1 \end{cases} \tag{3}$$

where  $w(T_{kit})$  is the weight assigned to term  $T_k$  from document  $D_i$  in html tag  $H_i$ . It is important to note that a term  $T_k$  can appear multiple times in multiple tags within a document. Thus, given that term  $T_k$  has a count  $C_{ik}$  in document  $D_i$ , the term frequency  $tf_{ik}$  in Equation (3) is computed as follows:

$$tf_{ik} = \sum_{tag} C_{ik} * w(T_{kit}) \tag{4}$$

The dissimilarity between the document vectors  $d_i$ , and query  $d_j$  is computed by measuring the angle between them, using the function defined in Equation (5). The computed pair-wise dissimilarities require further processing to get the actual individual document dissimilarities. The individual document dissimilarities are derived

using *k*-dissimilarity, neighborhood dissimilarity, and nearest dissimilarity density adapted from the local outlier concept [3], and defined next.

$$DIS \left( \vec{d}_i, \vec{d}_j \right) = 1 - \left( \frac{\sum_k w_{ik} w_{jk}}{\sqrt{\sum_k w_{ik}^2} \sqrt{\sum_k w_{jk}^2}} \right) \tag{5}$$

**Definition 1:** Let *d* be a document from a category *C*, let *k* be a natural number, and let *dis* be dissimilarity metric on *C*, then the *k*-dissimilarity of *d*, denoted *k*-*diss*(*d*), is the dissimilarity *dis*(*d*, *m*) between *d* and a document *m* ∈ *C*, such that at least for *k* documents *m*<sup>1</sup> ∈ *C*: it holds that *dis*(*d*, *m*<sup>1</sup>) < *dis*(*d*, *m*), and for at most *k*-1 document *m*<sup>1</sup> ∈ *C*: it holds that *dis*(*d*, *m*<sup>1</sup>) < *dis*(*d*, *m*). ■

The motive for computing *k*-dissimilarity of *d* is to determine which documents constitute the neighbors of *d*. It also helps in determining whether the neighborhood around *d* is dense or sparse. Any document with dissimilarity from *d* greater than *k*-dissimilarity of *d* cannot be considered a neighbor of *d*. The number of neighbors a document has may vary for documents, even if they have the same *k*.

**Definition 2:** The neighborhood dissimilarity denoted *Neigh<sub>d</sub>*(*d*) contains every document that has a dissimilarity not greater than *k*-dissimilarity of *d*. Non-trivial neighborhood dissimilarity consists of all the documents in the category. ■

**Definition 3:** The nearest dissimilarity density of a document *d*, denoted *Nearest<sub>d</sub>*(*d*) is defined as the ratio of the cardinality of neighborhood dissimilarity of *d* to the sum of the actual document dissimilarities within the neighborhood, and is given by:

$$Nearest_d(d) = \frac{\sum_{Diss \in Diss_d(d)} DIS}{card( Neigh_d(d))} \tag{6}$$

where DIS are the dissimilarities of documents within the dissimilarity neighborhood. The nearest dissimilarity density shows the distribution of document dissimilarities with respect to other documents. High nearest dissimilarity density means the neighborhood of the document is very dense, and hence has high potential of being an outlier. The nearest dissimilarity densities are sorted in descending order and the *top-n* documents with the highest nearest dissimilarity densities are declared outliers, where *n* is the number of outliers needed. The outliers produced are local with respect to the individual documents in the category.

## 4 Experimental Results and Analysis

The dataset for the experiment is the course and faculty corpus taken from the webkb dataset [18]. In this experiment we rely on *embedded motifs* to determine the effectiveness of HyCOQ algorithm. The experiment consists of embedding some faculty pages in the course pages and then checking how many of these are correctly identified. We test the effectiveness of HyCOQ with a baseline using full-words and n-grams with and without a dictionary. The running times of HyCOQ using different data types, first with a dictionary and then without one are also compared. Embedded motifs are used in this experiment because there are no benchmark data for testing web content outliers.



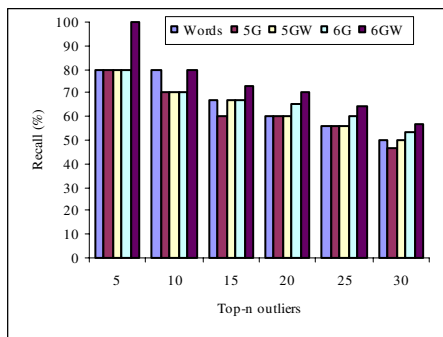


Fig. 3. Top-n Outliers without Dictionary

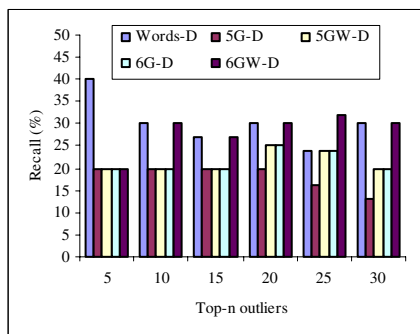


Fig. 4. Top-n Outliers with Dictionary

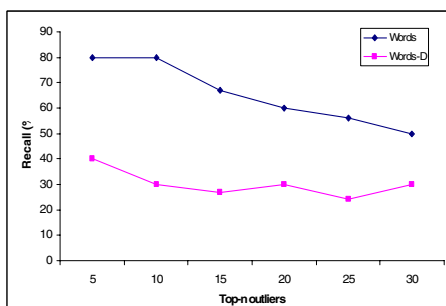


Fig. 5. Top-n Outliers using Words

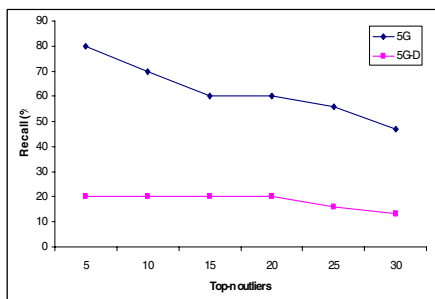


Fig. 6. Top-n Outliers using 5-grams

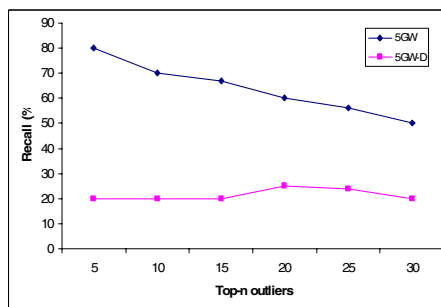


Fig. 7. Top-n Hybrid 5-gram Outliers

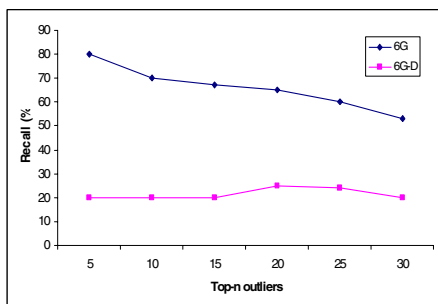


Fig. 8. Top-n Outliers with 6-grams

The effectiveness of HyCOQ is tested using 300 course pages and 30 faculty pages embedded in them as motifs to be identified. The rationale for choosing 30 faculty pages as embedded motifs is that outliers usually constitute less than 10% of the entire dataset [4]. The content is extracted and preprocessed. Hybrid document profiles are generated using 5-grams and 6-grams because experimental results show higher order n-grams are very effective in capturing similarities between document [5, 9]. The algo-

rithm is tested without a domain dictionary with words, 5-grams (5G), 6-grams (6G), 5-grams + words (5GW) and 6-grams+ words (6GW). The number of motifs correctly identified among the top- $n$  ( $n = 5, 10, 15, 20, 30$ ) outliers are shown in Figure 3. The experiment is repeated using a domain dictionary on the same data set. The dictionary is created from 50 course pages taken from the course corpus using simple random sampling without replacement. The 50 pages are excluded from the 300 pages used as our baseline dataset. The algorithm stops after computing document dissimilarities because every document is compared with the same dictionary, and hence does not produce pair-wise dissimilarities. The results obtained using words; 5-grams (5G-D), 6-grams (6G-D), 5-grams + words (5GW-D) and 6-grams + words (6GW-D) are shown in Figure 4. Figures 3 and 4 show mining web content outliers without a domain dictionary is more effective than with a domain dictionary. About 70% of the motifs are correctly identified without a dictionary compared to less than 40% with a domain dictionary, irrespective of the type of data used. Moreover, hybrid documents produce highest recall irrespective of whether a dictionary is used or not.

Figures 5 to 8 show the performance of HyCOQ with and without a domain dictionary for different data types. There is consistent improvement of over 100% when a dictionary is not used compared to using the dictionary. The maximum recall recorded for words, n-grams and hybrid without a domain dictionary is 80% compared to 40% for words with domain dictionary. The average recall is even higher reaching 100% at the top 1.5% when a hybrid of words and 6-grams are used without a dictionary as shown in Figure 9. Finally, Figure 10 reveals using a hybrid document achieves a better recall than using either the raw n-grams or words.

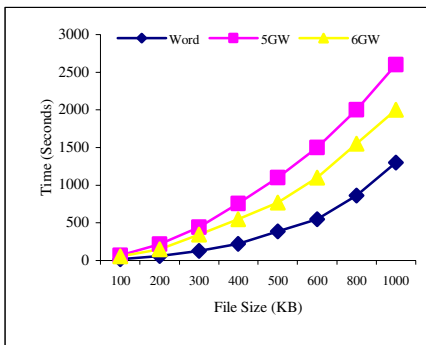


Fig. 9. Top- $n$  Hybrid 6-gram Outliers

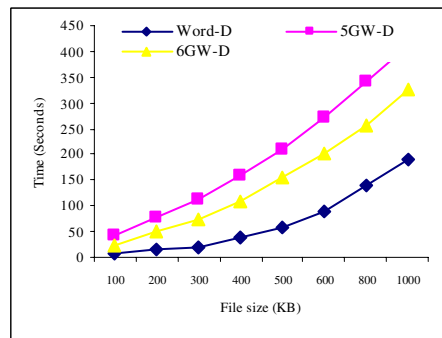


Fig. 10. Top- $n$  Outliers: Words, GG & 6GW

## 4.2 Testing for Response Time

The response time dataset consists of 1000 web pages from the course corpus of *webkb* dataset [18]. The HyCOQ algorithm is applied using words, 5GW and 6GW without a domain dictionary. The response times are recorded for different data size as shown in Figure 11. Each 100KB data consist of approximately 100 processed files. The experiment is repeated on the same dataset but this using a domain dictionary with the results depicted in Figure 12. The overall response time for using words

alone is better than using either of the hybrids (5GW and 6GW) irrespective of whether or not a dictionary is used. Among the hybrids, 6GW has a better response time than 5GW as shown in Figures 11 and 12.

Though the overall response time of using a dictionary is better than not using it, the quality of the results returned cannot be under estimated. In outlier mining where the entire data contains very few outliers, the quality of the results is very important. Thus, using the hybrid dataset without a domain dictionary with a recall of about 70% is better than using a domain dictionary with an average recall of less than 40% which cannot even be guaranteed because the results depend on the quality of the dictionary.

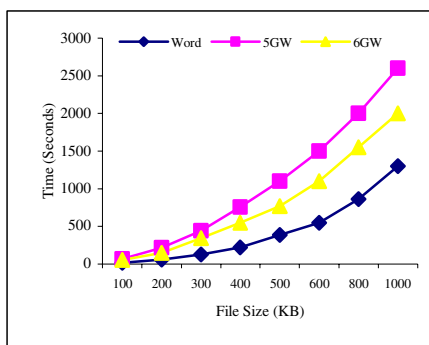


Fig. 11. Response Time without Dictionary

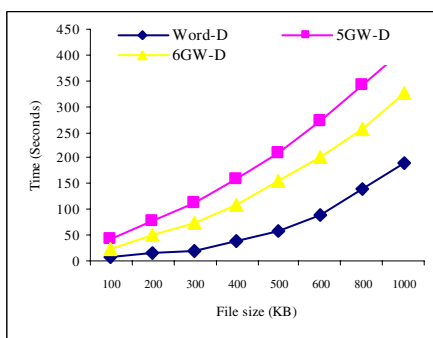


Fig. 12. Response Time with Dictionary

## 5 Summary and Conclusions

This paper proposes the HyCOQ algorithm for mining web content outliers using hybrid documents as data input. The proposed algorithm has two major advantages over its predecessor: 1) it does not require a domain dictionary, hence eliminating the need for a domain expert; 2) the use of hybrid documents eliminates the weaknesses associated with n-gram and word-based systems. The experimental results with embedded motifs show using the hybrid documents especially 6GW (6-grams + words) performs better (recall) than using the words or the n-grams alone irrespective of whether a dictionary is used or not. The overall results indicate using any data type (i.e., words, n-grams, or hybrids) without a dictionary has a better recall than using the same dataset with a domain dictionary. The results show an average of about 70% recall without a dictionary compared to less than 40% with domain dictionary. There is a remarkable improvement of more than 100% when domain dictionary is not used compared to when it is used. However, the response time reported for the same dataset with a domain dictionary is more promising than without a domain dictionary.

Though the overall response time of using a dictionary is better than without it, the quality of results returned cannot be under estimated. In an area like outlier mining where the entire data contains very few outliers, the quality of the results is very important. Thus, using the hybrid dataset without a domain dictionary with a recall of

about 70% is better than using a domain dictionary with average recall less than 40%. We note here that even the 40% recall is not guaranteed and depends very much on the quality of the dictionary as opposed to the data being mined.

## References

1. Agyemang M., Barker K., and Alhaji R.: Framework for Mining Web Content Outliers. *Proc. of ACM SAC*, Cyprus, pp.590-594, 2004.
2. Agyemang M., Barker K., and Alhaji R.: Mining Web Content Outliers using Structure Oriented Weighting Techniques and N-grams. *Proc. of ACM SAC*, New Mexico, 2005.
3. Breunig M.M., Kriegel H-P., Ng R.T., and Sander J.: LOF: Identifying Outliers in Large Dataset, *Proc. of ACM SIGMOD*, Dallas, TX, pp.93-104, 2000.
4. Barnett V. and Lewis T., Outliers in Statistical Data, John Wiley, 1994.
5. Cavnar B.W. and Trenkle M.J.: N-Gram-Based Text Categorization, *Proc. of SDAIR*, 1994.
6. Chakrabarti S., et al: Mining the Link Structure of the World Wide Web. *IEEE Computer* 32(8): 60-67, 1999.
7. Chakrabarti S.: Data Mining for Hypertext- A Tutorial Survey. *ACM SIGKDD Explorations*, 1(2):1-11, 2000
8. Cooley R., Mobasher B., and Srivastava J.: Information and Pattern Discovery on the World Wide Web, *SIGKDD Explorations*, 1(2), 2000.
9. Damashek M.: Gauging Similarity with N-Grams: Language Independent Categorization of Text. *Science*, 267, pp.843-848, 1995.
10. Etzioni O.: The World Wide Web: Quagmire or Gold Mine, *Communication of the ACM*, 39(11):65-68, 1996.
11. Jin W., Tung A. K-H., and Han J.: Mining Top-n Local Outliers in Large Databases, *Proc. of ACM SIGKDD*, San Francisco, CA, pp.293-298, 2001.
12. Johnson T., Kwok I., and Ng R.: Fast Computation of 2-D depth Contours, *Proc. of ACM SIGKDD*, pp.224-228, 1998.
13. Knorr E.M., Ng R.T.: A Unified Notion of Outliers: Properties and Computation. *Proc. of KDD*, pp 219-222, 1997.
14. Knorr E.M. and Ng R.T.: Algorithms for Mining Distance-Based Outliers in Large Dataset, *Proc. of VLDB*, New York, pp.392-403, 1998.
15. Liu B., Ma Y., and Yu P.S.: Discovering Unexpected Information from Your Competitors' Web Sites, *Proc. of ACM SIGKDD*, San Francisco, CA, pp.144-153, 2001.
16. Ramaswamy S., Rastogi R., and Shim K.: Efficient Algorithms for Mining Outliers from Large Data Set, *Proc. of ACM SIGMOD*, pp.427-438, 2000.
17. Salton G. and Buckley C.: Term-Weighting Approaches in automatic Text Retrieval. *Information Processing and Management*. 24(5) 513-523, 1988.
18. <http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data>, Dec. 2004.
19. [http://ir.dcs.gla.ac.uk/resources/linguistic\\_utils/stop\\_words](http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words), Dec. 2004.

# Incremental Data Mining Using Concurrent Online Refresh of Materialized Data Mining Views

Mikołaj Morzy, Tadeusz Morzy, Marek Wojciechowski, and Maciej Zakrzewicz

Institute of Computing Science,  
Poznań University of Technology,  
Piotrowo 3A, 60-965 Poznań, Poland  
{mmorzy, tmorzy, mwojciechowski, mzakrzewicz}@cs.put.poznan.pl

**Abstract.** Data mining is an iterative process. Users issue series of similar data mining queries, in each consecutive run slightly modifying either the definition of the mined dataset, or the parameters of the mining algorithm. This model of processing is most suitable for incremental mining algorithms that reuse the results of previous queries when answering a given query. Incremental mining algorithms require the results of previous queries to be available. One way to preserve those results is to use materialized data mining views. Materialized data mining views store the mined patterns and refresh them as the underlying data change.

Data mining and knowledge discovery often take place in a data warehouse environment. There can be many relatively small materialized data mining views defined over the data warehouse. Separate refresh of each materialized view can be expensive, if the refresh process has to re-discover patterns in the original database. In this paper we present a novel approach to materialized data mining view refresh process. We show that the concurrent on-line refresh of a set of materialized data mining views is more efficient than the sequential refresh of individual views. We present the framework for the integration of data warehouse refresh process with the maintenance of materialized data mining views. Finally, we prove the feasibility of our approach by conducting several experiments on synthetic data sets.

## 1 Introduction

Data mining, or knowledge discovery in databases, is a non-trivial process of finding valid, novel, useful, and ultimately understandable patterns and regularities in very large data volumes [5]. Data mining systems are quickly evolving from specific to general-purpose systems that are tightly coupled with the existing relational database technology. Integration is usually performed in the data warehouse, which serves high quality data to various mining techniques. Mining processing characteristics differs significantly from typical database workload. Hence, new methods of data mining query processing and optimization are being developed. One of these methods is incremental discovery of patterns.

The patterns discovered as the result of the execution of a data mining algorithm can be regarded as an answer to a sophisticated database query. A user defines the set of mined data using standard SQL commands and determines the parameters governing a given data mining algorithm. In response, relevant patterns are returned to the user for evaluation. Users usually do not achieve satisfying results immediately. It is an iterative process, where in each consecutive step the user evaluates the patterns and, suitably to the needs, expectations, and experience, modifies either the mined dataset, or algorithm parameters, or both. Because of this iterative and repetitive nature of mining processing, a data mining system must efficiently exploit the results of previous queries when fulfilling user requests. Data warehouses facing similar requirements in on-line analytical processing are materializing the results of queries as snapshots and rewrite incoming queries to use the materialized data. The same principle applies to data mining systems, where previously discovered patterns are stored in materialized data mining views and used to efficiently answer user queries.

The main problem in using materialized patterns gathered during mining is the freshness of the patterns. Each update of the source data can potentially invalidate some or all patterns stored in a materialized data mining view. This is particularly important in a data warehouse, where refresh of the source data happens regularly, often on a daily or even hourly basis. To cope with this problem, incremental mining techniques were proposed that aim at efficient maintenance of materialized patterns by running the base algorithm only on the difference set, and minimizing the number of full data reads necessary to validate the patterns. The main drawback of this approach is the fact, that the proposed methods refresh materialized data mining views separately, disregarding the properties of the data warehouse environment.

The refresh cycle of modern data warehouses becomes more and more frequent. As the result, the volume of the data loaded upon each refresh becomes relatively smaller as compared to the size of the entire data warehouse. On the other hand, the number of materialized data mining views defined in a data warehouse increases. In practice, materialized data mining views that span entire fact tables are rare. Instead of having a single large materialized data mining view, users define several well-focused views that cover a specific area of analysis (e.g., buyers of a particular product, purchases made during particular time of day, etc.). Usually, only a small subset of a huge fact table is used as the data source for such materialized data mining view. Consequently, the probability that a given tuple contained in the data loaded into a data warehouse upon refresh is likely to influence the patterns stored in a materialized data mining view is much smaller than assumed in traditional approaches. In such circumstances, using the entire load of new data and repeating incremental maintenance procedures for all materialized data mining views separately is a waste of resources.

We propose to tackle the problem of materialized data mining view maintenance from another perspective. Our solution is concurrent online refresh of a set of materialized data mining views. In our approach the maintenance of materialized data mining views becomes an integral part of the data warehouse

refresh process. When a tuple is loaded into the warehouse, materialized data mining views that could become affected by that tuple are updated simultaneously. In this paper we present a framework for online maintenance of a set of materialized data mining views. We argue that our assumptions are reasonable with respect to practical data warehouse implementations. Our contribution is the following. We demonstrate the structure for fast lookup of candidate materialized data mining views that could become affected by an insert of a tuple. We present a novel algorithm for concurrent online incremental mining and we experimentally prove the feasibility of the proposed approach by comparing our algorithm with other algorithms. Experimental comparison of our algorithm with two algorithms proposed in the literature so far (Apriori and IUA) shows that our algorithm outperforms previous proposals.

The remainder of the paper is organized as follows. Following this paragraph we present the related work and definitions of basic terms used throughout the paper. The idea of using materialized results of previous data mining queries to answer subsequent user requests is presented in Section 2. In Section 3 we present a novel approach to the materialized pattern maintenance problem which consists in concurrent online refresh of materialized data mining views. Experimental evaluation of our approach is presented in Section 4. The conclusions are contained in Section 5.

## 1.1 Related Work

The problem of association rule mining was introduced in [2]. The paper identified the discovery of frequent itemsets as a key step in association rule mining. In [3] the authors presented basic algorithm called *Apriori*, which quickly became the seed of several other data mining algorithms. The first algorithm for maintaining discovered association rules using incremental technique, called FUP, was proposed in [4]. In [13] a new Incremental Updation Algorithm (IUA) was proposed. IUA minimized the number of full database scans to discover association rules in the updated data using the idea of the negative border of the collection of frequent itemsets [7].

The work on materialized views started in the 1980s. Multiple algorithms for view maintenance were developed [12]. Further research led to the creation of cost models for materialized view maintenance and applying views to enforce integrity constraints in databases. A summary of view maintenance techniques can be found in [6]. Materialized data mining views were first proposed in [10], and quickly became an important tool in data mining query optimization [8,9,14]. To the best of our knowledge the idea of concurrent online refresh of a set of materialized data mining views has not been presented yet.

## 1.2 Basic Definitions

Let  $I = \{i_1, \dots, i_n\}$  be a set of literals called items. Let  $D$  be a set of variable length transactions and  $\forall T \in D : T \subseteq I$ . We say that the transaction  $T$  supports an item  $x$  if  $x \in T$ . We say that the transaction  $T$  supports an itemset  $X$

if it supports every element  $x \in X$ . The *support* of an itemset is the number of transactions supporting the itemset. The problem of discovering frequent itemsets consists in finding all itemsets with the support higher than user-defined minimum support threshold denoted as *minsup*. An itemset with the support higher than *minsup* is called a *frequent itemset*. Given a collection of frequent itemsets  $L$ . The negative border  $NBd(L)$  of the collection  $L$  consists of all sets  $s_i$ , such that  $s_i \notin L \wedge \forall s'_i \subset s_i : s'_i \in L$ .

An association rule is an implication of the form  $X \rightarrow Y$  where  $X \subset I, Y \subset I$  and  $X \cap Y = \emptyset$ .  $X$  is called the *body* of the rule whilst  $Y$  is called the *head* of the rule. Two measures represent statistical significance and strength of a rule. The *support* of a rule is the number of transactions that support  $X \cup Y$ . The *confidence* of a rule is the ratio of the number of transactions that support the rule to the number of transactions that support the head of the rule. The problem of discovering association rules consists in finding all rules with support and confidence higher than the user-specified thresholds of minimum support and confidence, called *minsup* and *minconf* respectively.

## 2 Data Mining Using Materialized Views

MineSQL [11] is a multi-purpose data mining query language which uses data mining queries to express data mining tasks. The syntax of MineSQL resembles standard SQL and provides excellent means of integration of data mining requests with the underlying database management system. MineSQL allows to issue commands that discover frequent itemsets, association rules, and sequential patterns. MineSQL uses additional data types (e.g. SET, ITEMSET, RULE) as well as operators and functions for those data types (e.g. CONTAINS, BODY(x), HEAD(x)). The following data mining query discovers all association rules with support higher than 2.5% and confidence higher than 70%, which contain an item 'Bordeaux Pomerol' in the body of the rule. Mining is performed over transactional data on premium customers for the 2nd half of the year 2004 .

```
MINE RULE r, HEAD(r), BODY(r)
FOR products FROM (
  SELECT SET(product) AS products
  FROM PurchaseFacts
  WHERE time_id >= '01.07.2004'
     AND time_id <= '31.12.2004'
     AND customer_type = 'Premium'
  GROUP BY transaction_id )
WHERE SUPPORT(r) > 0.025
     AND CONFIDENCE(r) > 0.7
     AND BODY(r) CONTAINS TO_SET('Bordeaux Pomerol');
```

In traditional databases a view defines a mapping function from a set of base relations to the derived relation. The function is computed upon each reference to the view. Views hide complex data structures from a user and provide an additional independence layer between an application and the underlying database



schema. Changes occurring in the database schema are reflected only in the view definition with no impact on the end-user application. In order to avoid computational overhead, the contents of the view can be materialized in the database. The materialized copy of the data can be quickly accessed, thus bypassing expensive computation of the view. Data stored in a materialized view are not automatically refreshed when base relations change. Therefore, view maintenance techniques are necessary to reflect changes that occur in base relations of a materialized view. Often, modifications of base relations affect only a part of the materialized view. Incremental view maintenance techniques avoid recomputation of the entire view contents by determining the parts of the materialized view that should be updated.

Similarly to traditional database views, data mining views can be used to simplify application development, hide the complexity of data mining algorithms behind standard view interface, and enable incremental mining techniques by materializing results of data mining queries for further processing. Consider the following MineSQL statement that defines a materialized data mining view `v_saint_emilion`.

```
CREATE MATERIALIZED VIEW v_saint_emilion
REFRESH 7 AS
MINE RULE r, BODY(r), SUPPORT(r), CONFIDENCE(r)
FOR products FROM (
  SELECT SET(product) AS products
  FROM PurchaseFacts
  WHERE time_id >= '01.01.2004'
     AND time_id <= '31.12.2004'
  GROUP BY transaction_id
  HAVING AVG(price) >= 10 )
WHERE SUPPORT(r) > 0.025
  AND HEAD(r) CONTAINS TO_SET('Saint Emilion');
```

The definition of the view contains two classes of constraints: *database constraints* appear within the `WHERE` clause in the `SELECT` subquery, whereas *mining constraints* appear within the `WHERE` clause in the `MINE` statement. Database constraints delimit the part of the database that constitutes the source dataset. Mining constraints define patterns that are interesting to the user. Materialized data mining view not only separates the user from the technical details of the underlying mining algorithm, but provides the storage for discovered patterns. Every pattern in a materialized data mining view has a timestamp representing its creation time and validity period. One can provide the `REFRESH` clause that defines the period after which the contents of the materialized data mining view should be refreshed. Materialized views can be refreshed manually or automatically. The refresh of a materialized view could be performed by an incremental mining algorithm, or could involve the recomputation of the entire view.

The importance of materialized data mining views stems from the fact that the contents of the materialized data mining view can be used to efficiently answer a data mining query which is similar to the materialized view definition.

Depending on the relations between a query and the view definition, several different mining methods are available. These methods include incremental mining, complementary mining, verifying mining, and full mining. Data mining query optimization using materialized data mining views is covered in [8,14].

### 3 Concurrent Online Refresh of Materialized Data Mining Views

Data warehouse is an integrated collection of high-quality data supporting decision making. Based on this data, users can define multiple, possibly overlapping, collections of related data that serve as data sources for data mining queries. We argue that in typical applications users perform a focused selection of source data of interest and constraint their data mining activities to the selected subsets of the original data. We attribute this behavior to the fact that very large volumes of data produce patterns that are too general to be useful in analysis and decision making. Rather, users concentrate on smaller sets of data that are relevant to a given data mining query. After determining the subset of interesting data and setting the parameters of a mining algorithm, users can store their mining activity as a materialized data mining view and decide on the refresh frequency. Therefore, one should perceive a data warehouse as an environment for multiple different materialized data mining views that can be refreshed and maintained independently. Moreover, as changes to the data warehouse do not happen continuously over time, but are loaded in chunks during periodical data warehouse refresh, the refresh of materialized data mining views can be integrated with the process of the entire data warehouse refresh. It is worth noticing that during data warehouse refresh, when new tuples are loaded into base tables, they do not necessarily invalidate all materialized data mining views. Whether a tuple invalidates the patterns stored in a materialized data mining view or not, depends solely on the definition of the materialized data mining view, in particular, on the database constraints of the view.

In our implementation we are using a special index table to store the definitions of materialized data mining views. For all database tables that are used as the source for data mining views, table attributes are mapped to columns in the index table. Each materialized data mining view is described as a single row in the index table. In addition to columns representing attributes of relational tables, the index table stores the thresholds of minimum support and confidence provided in the view definition. If the definition of the materialized data mining view contains a database constraint defined on a base table attribute, this fact is reflected in the index table by inserting the relational operator ( $=$ ,  $\leq$ ,  $\geq$ ,  $\neq$ , etc.) with the associated constant value into the appropriate column of the index table. A special symbol '\*' is used if attributes of a base table are used in the materialized data mining view without any constraints. The index table is used to quickly decide, which of the materialized data mining views defined in the data warehouse are affected by the insertion of a given tuple. This is done by comparing the values in the inserted tuple with the constants stored in the appropriate attributes of the index table.

Each materialized data mining view is implemented as a relational table. The table contains both frequent itemsets constituting the answer to the data mining query, and the negative border of the collection of frequent itemsets. Each itemset is represented as a single row having two attributes: a numerical attribute containing the support of the itemset and a collection of items implemented as a nested table of varying length. This schema can be easily extended to support the storage of association rules. Given an association rule  $X \rightarrow Y$ . Both  $X$  and  $Y$  must be frequent itemsets. Therefore, both itemsets appear in the base relational table. The row representing the itemset  $X$  has an additional column **HEADS** which is a nested table of **head** objects. Each **head** object consists of the numerical confidence measure and the pointer to the itemset forming the head of the rule (in the above example the pointer points to the location of the itemset  $Y$ ). Analogously, every itemset has an additional column **BODIES** implemented as a nested table of **body** objects. Again, each **body** object consists of the confidence measure of the rule and the pointer to the itemset forming the body of the rule. Pointers are either artificial primary keys or physical row addresses. Using bi-directional pointers to rule elements allows for fast lookup of rules containing a given itemset in the body or the head of the rule.

The algorithm for concurrent online refresh of materialized data mining views, denoted OUA for Online Updation Algorithm, proceeds as follows. The insertion of new tuples into the data warehouse base table triggers the verification procedure. First, definitions of all materialized data mining views defined on the updated base table are retrieved from the index table. Next, values of attributes of a newly inserted tuple are compared to the values of attributes used in database constraints of materialized data mining views. The comparison is performed using a special function which selects appropriate relational operators to test, whether the tuple satisfies all constraints defined in the definition of a materialized data mining view. If the comparison succeeds, the procedure updates the view.

Given a materialized data mining view  $MDMV$  and a newly inserted tuple  $t$  that affects the view. Let  $L$  denote the collection of frequent itemsets present in the materialized data mining view  $MDMV$ . Let  $NBd(L)$  denote the negative border of the collection of frequent itemsets  $L$  and let  $C_k$  denote the collection of candidate itemsets of the size  $k$ .

In the first step the set of items contained in the newly inserted tuple  $t$  is divided to form the collection of one-element candidate itemsets  $C_1$ . For all elements in  $C_1$  the algorithm searches for itemsets  $s \in L \cup NBd(L)$ , such that  $s = c$ , and increases the support count of these itemsets. If, during this step, an itemset from the negative border  $NBd(L)$  becomes frequent, it is moved to  $L$  and the negative border is expanded to reflect this move. Next, itemsets  $C_1 \cap L$  are used to generate the set of candidate 2-itemsets  $C_2$ . Observe that only 2-itemsets contained in the inserted tuple are used to grow  $L$  and  $NBd(L)$ . This procedure repeats until no more candidate  $k$ -itemsets can be generated (candidates are generated using standard *apriori-gen* procedure of the *Apriori* algorithm). After processing all new tuples the algorithm checks if the negative

border of the collection of frequent itemsets should be updated. This happens if there is a set that moved from the negative border to the collection of frequent itemsets. This step may require a full database scan.

*Example 1.* Given the materialized view  $MDMV$  with  $L = \{A, B, C, AB, AC\}$  and  $NBd(L) = \{BC\}$ . Let the newly inserted tuple  $t = \langle A, B, C \rangle$ . First, all elements of the tuple  $t$  are used to create the collection of candidate 1-itemsets  $C_1 = \{A, B, C\}$ . This collection is compared with  $L \cup NBd(L)$  and the appropriate support counts are incremented. No itemsets are moved from the negative border to the set of frequent itemsets. Next, the set  $L_1$  is determined as  $L_1 = C_1 \cap L = \{A, B, C\}$ . These itemsets are used to generate the set of candidate 2-itemsets  $C_2 = \{AB, AC, BC\}$ . Again, these itemsets are compared with  $L \cup NBd(L)$  to increase appropriate support counts. As the result, the itemset  $BC$  is moved from  $NBd(L)$  to  $L$  and the negative border  $NBd(L)$  is expanded with the itemset  $ABC$ . As in previous step, the set  $L_2$  is determined as  $L_2 = C_2 \cap L = \{AB, AC, BC\}$ . This procedure repeats until no new candidates can be generated. Support counts for itemsets from the expanded negative border are determined during additional database scan.

## 4 Experimental Results

All experiments were conducted on Dell Pentium M 1,4GHz with 768MB of RAM running Windows 2000 and Oracle 9i. Data sets were created using DB-Gen generator from the Quest Project [1]. Original database contained 100 000 transactions, the average size of the transaction was 40, and the number of different items was set to 100 000. For comparison we have chosen the basic Apriori algorithm (no incremental mining at all) and Incremental Updation Algorithm [13]. The size of the base table update varied from 500 to 5000 new transactions (i.e., from 0.5% to 5% of the original data volume). The percentage of the original base table covered by the materialized data mining view varied from 2.5% to 50%, the support threshold changed from 1.5% to 5%, the number of materialized data mining views that were simultaneously updated varied from 5 to 20.

The results of experiments are depicted in Figures 1-4. As expected, our algorithm works best when the number of materialized data mining views is large and the degree of coverage of base table is small. Again, we argue that this situation is typical for most applications using data mining techniques within the data warehouse environment. An important factor that affects the performance of our algorithm is the size of the update. For larger updates the cost of processing of each tuple separately surpasses the gain of not reading the update several times (especially when the number of concurrently updated materialized views is small). In such cases Incremental Updation Algorithm is a winner. We believe that this result is not discouraging, because we are observing a continuous shrinking of the data warehouse refresh window. Our algorithm is best suited for frequently refreshed warehouses, where the contents of the data warehouse must be synchronized with operational databases on a daily or hourly basis.

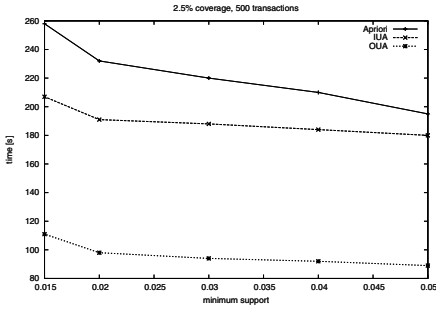


Fig. 1. time vs. support, 2.5% coverage

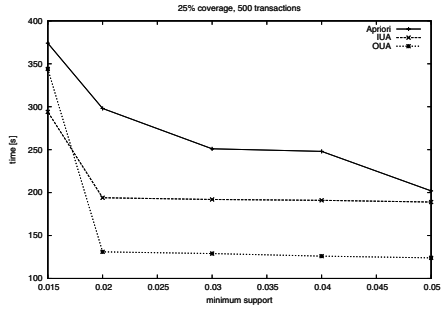


Fig. 2. time vs. support, 25% coverage

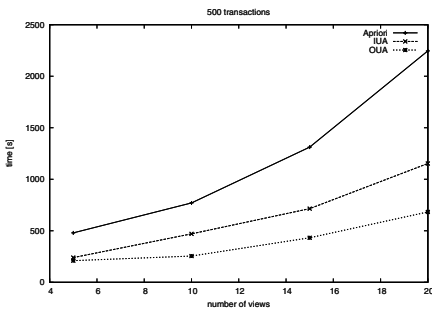


Fig. 3. time vs. number of views

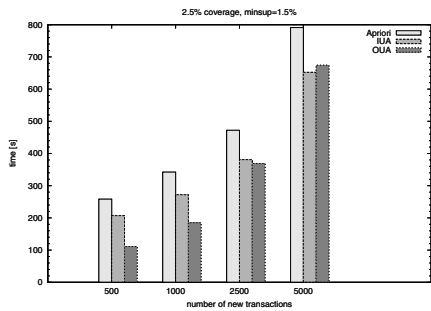


Fig. 4. time vs. update size

## 5 Conclusions

In this paper we have presented Online Updating Algorithm, which implements a novel approach to materialized data mining view maintenance problem. Instead of performing separate refresh of a set of materialized data mining views we propose to update them simultaneously, during the data warehouse refresh process. Our algorithm outperforms previously proposed methods in environments where many materialized data mining views are defined over relatively small subsets of source data. We argue that this assumption holds in most practical applications, hence our algorithm provides improvement over other approaches.

## References

1. R. Agrawal, M. J. Carey, C. Faloutsos, S. P. Ghosh, M. A. W. Houtsma, T. Imielinski, B. R. Iyer, A. Mahboob, H. Miranda, R. Srikant, and A. N. Swami. Quest: A project on database mining. In R. T. Snodgrass and M. Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, page 514, Minneapolis, Minnesota, may 1994. ACM Press.

2. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.
4. D. W.-L. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In S. Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 106–114. IEEE Computer Society, 1996.
5. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
6. A. Gupta and I. S. Mummick. *Materialized Views: Techniques, Implementations, and Applications*. The MIT Press, 1999.
7. H. Mannila and H. Toivonen. An algorithm for finding all interesting sentences. In *Proc. of the 6th Int'l Conference on Database Theory*, pages 215–229, 1996.
8. M. Morzy, T. Morzy, and Z. Królikowski. Incremental association rule mining using materialized data mining views. In T. Yakhno, editor, *Advances in Information Systems, 3rd International Conference, ADVIS 2004, Izmir, Turkey, October 20-22, 2002, Proceedings*, volume 3261 of *Lecture Notes in Computer Sciences*, pages 77–87. Springer-Verlag, 2004.
9. M. Morzy, M. Wojciechowski, and M. Zakrzewicz. Cost-based sequential pattern query optimization in presence of materialized results of previous queries. In M. A. Kłopotek, S. Wierzchon, and M. Michalewicz, editors, *Intelligent Information Systems 2002*, Advances in Soft Computing, pages 435–444. Physica-Verlag, jun 2002.
10. T. Morzy, M. Wojciechowski, and M. Zakrzewicz. Materialized data mining views. In D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, editors, *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000, Lyon, France, September 13-16, 2000, Proceedings*, volume 1910 of *Lecture Notes in Computer Science*, pages 65–74. Springer, 2000.
11. T. Morzy and M. Zakrzewicz. Sql-like language for database mining. In *1st East European Symposium on Advances in Databases and Information Systems, ADBIS 1997, St.Petersburg, Russia, September , 1997, Proceedings*, 1997.
12. N. Roussopoulos. Materialized views and data warehouses. *SIGMOD Record*, 27(1), 1998.
13. S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka. An efficient algorithm for the incremental update of association rules in large databases. In D. Heckerman, H. Mannila, and D. Pregibon, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, pages 263–266. AAAI Press, 1997.
14. M. Zakrzewicz, M. Morzy, and M. Wojciechowski. A study on answering a data mining query using a materialized view. In C. Aykanat, T. Dayar, and I. Korpeoglu, editors, *Computer and Information Sciences - ISCIS 2004, 19th International Symposium, Antalya, Turkey, October 27-29, 2003, Proceedings*, volume 3280 of *Lecture Notes in Computer Sciences*, pages 493–502. Springer-Verlag, 2004.

# A Decremental Algorithm for Maintaining Frequent Itemsets in Dynamic Databases\*

Shichao Zhang<sup>1,3</sup>, Xindong Wu<sup>2</sup>, Jilian Zhang<sup>3</sup>, and Chengqi Zhang<sup>1</sup>

<sup>1</sup> Faculty of Information Technology, University of Technology Sydney, Australia  
{zhangsc, chengqi}@it.uts.edu.au,

<sup>2</sup> Department of Computer Science, University of Vermont, USA  
xwu@cs.uvm.edu

<sup>3</sup> Department of Computer Science, Guangxi Normal University, China  
zhangjilian@yeah.net

**Abstract.** Data mining and machine learning must confront the problem of pattern maintenance because data updating is a fundamental operation in data management. Most existing data-mining algorithms assume that the database is static, and a database update requires rediscovering all the patterns by scanning the entire old and new data. While there are many efficient mining techniques for data additions to databases, in this paper, we propose a decremental algorithm for pattern discovery when data is being deleted from databases. We conduct extensive experiments for evaluating this approach, and illustrate that the proposed algorithm can well model and capture useful interactions within data when the data is decreasing.

## 1 Introduction

The dynamics of databases can be represented in two aspects: (1) content updates over time and (2) incremental size changes. When some transactions of a database are deleted or modified, the content of the database has been updated. This database is referred to as an *updated database* and mining with the updated database is referred to as *decremental mining*. When some new transactions are inserted or appended into a database, the size of the database has been changed. This database is referred to as an *incremental database* and mining with the incremental database is referred to as *incremental mining*. This paper investigates the issue of mining updated databases<sup>1</sup>.

When a database is updated on a regular basis, running a data-mining program all over again each time when there is an update might produce significant computation and I/O loads. Hence, there is a need for data-mining algorithms to perform frequent pattern maintenance on incrementally updated databases without having to run the entire mining algorithm again [PZOD99]. This leads to many efficient mining techniques for data additions to databases, such as the FUP algorithm [CHNW96], ISM

---

\* This work is partially supported by large grants from the Australian Research Council (DP0449535 and DP0559536), a China NSFC major research program (60496321), and a China NSFC grant (60463003).

<sup>1</sup> For simplicity, this paper deals with deletions only, because a modification can be implemented as a deletion followed by an insertion.

[PZOD99], negative-border based incremental updating [TBAR97], incremental induction [U94] and the weighting model [ZZY03].

Unfortunately, no research efforts have been reported on pattern discovery when data is deleted from databases. Indeed, *delete* is one of the most frequently used operations in many DBMS systems, such as IBM DB2, MS SQL SERVER, and ORACLE. Usually, these DBMS systems use log files to record the committed changes in order to maintain the database consistency. Therefore we can easily obtain the data that is deleted from the original database by using the *delete* operation.

To solve the decrement problem, one can simply re-run an association rule mining algorithm on the remaining database. However, this approach is very inefficient, without making use of the previous computation that has been performed. In this paper, we propose an algorithm **DUA** (*Decrement Updating Algorithm*) for pattern discovery in dynamic databases when data is deleted from a given database. Experiments show that **DUA** is 2 to 13 times faster than re-running the Apriori algorithm on the remaining database. The accuracy of **DUA**, namely, the ratio of the frequent itemsets found by **DUA** in the remaining database over the itemsets found by re-running Apriori, is more than 99%.

The rest of the paper is organized as follows. We provide the problem description in Section 2. In Section 3, the DUA algorithm is described in detail. Experimental results are presented in Section 4. Finally, conclusions of our study are given in Section 5.

## 2 Problem Statement

### 2.1 Association Rule Mining

Let  $I = \{i_1, i_2, \dots, i_N\}$  be a set of  $N$  distinct literals called *items*, and  $DB$  be a set of variable length transactions over  $I$ , where transaction  $T$  is a set of items such that  $T \subseteq I$ . A transaction has an associated unique identifier called *TID*. An *itemset*  $X$  is a set of items, i.e., a subset of  $I$ . The number of items in an itemset  $X$  is the length of the itemset.  $X$  is called a *k-itemset* if the length of  $X$  is  $k$ . A transaction  $T$  contains  $X$  if and only if  $X \subseteq T$ . An association rule is an implication of the form  $X \rightarrow Y$ , where  $X, Y \subset I$ ,  $X \cap Y = \emptyset$ .  $X$  is the *antecedent* of the rule, and  $Y$  is the *consequent* of the rule. The support of a rule  $X \rightarrow Y$ , denoted as  $supp(X \cup Y)$ , is  $s$  if  $s\%$  transactions in  $DB$  contain  $X$ . and the confidence of rule  $X \rightarrow Y$ , denoted as  $conf(X \rightarrow Y)$ , is  $c$  if  $c\%$  transactions in  $DB$  that contain  $X$  also contain  $Y$ . That is,  $conf(X \rightarrow Y) = supp(X \cup Y) / supp(X)$ .

The problem of mining association rules from database  $DB$  is to find out all the association rules whose support and confidence are greater than or equal to the minimum support (*minsupp*) and the minimum confidence (*minconf*) specified by the user respectively. Usually, an itemset  $X$  is called frequent, if  $supp(X) \geq minsupp$ , and  $X$  is called infrequent if  $supp(X) < minsupp$ . The first step of association rule mining is to generate *frequent* itemsets using an algorithm like Apriori [AR94] or the FP-Tree [HPY00]. The second step then generates association rules based on the discovered *frequent* itemsets. This second step is straightforward, so the main problem of mining association rules is to find out all frequent itemsets in  $DB$ .



## 2.2 Updated Databases

The update operations include deletions and modifications on databases. Consider a transaction database  $TD = \{\{A, B\}; \{A, C\}; \{A, B, C\}; \{B, C\}; \{A, B, D\}\}$  where the database has several transactions, separated by a semicolon, and each transaction contains several items, separated by a comma.

The update operation on  $TD$  can basically be

**Case-1.** Deleting transactions from the database  $TD$ . For example, after deleting transaction  $\{A, C\}$  from  $TD$ , the updated database is  $TD1$  as

$$TD1 = \{\{A, B\}; \{A, B, C\}; \{B, C\}; \{A, B, D\}\}$$

**Case-2.** Deleting specified attributes from all transactions in the database  $TD$ . For example, after deleting  $B$  in  $TD$ , the updated database is  $TD2$  as

$$TD2 = \{\{A\}; \{A, C\}; \{A, C\}; \{C\}; \{A, D\}\}$$

**Case-3.** Updating a specified attribute with another attribute in all transactions. For example, after updating the attribute  $C$  to  $E$  in  $TD$ , the updated database is  $TD3$  as

$$TD3 = \{\{A, B\}; \{A, E\}; \{A, B, E\}; \{B, E\}; \{A, B, D\}\}$$

**Case-4.** Modifying a transaction in the database  $TD$ . For example, after modifying the transaction  $\{A, C\}$  to  $\{A, C, D\}$  for  $TD$ , the updated database is  $TD4$  as

$$TD4 = \{\{A, B\}; \{A, C, D\}; \{A, B, C\}; \{B, C\}; \{A, B, D\}\}$$

Mining updated databases generates a significant challenge: the maintenance of their patterns. To capture the changes of data, for each time of updating a database, we can possibly re-mine the updated database, but this would be a time-consuming procedure. In particular, when a database to be mined is very large and the changed content of each updating transaction is relatively small, re-mining the database is not an intelligent strategy. Our strategy for mining in updated databases is decremental discovery. To simplify the description, this paper focuses on the above Case-1 and Case-2.

## 2.3 Maintenance of Association Rules

Let  $DB$  be the original transaction database,  $db$  be a dataset randomly deleted from  $DB$ , and  $DB-db$  be the remaining database.  $|DB|$ ,  $|db|$ , and  $|DB-db|$  denote the size of  $DB$ ,  $db$ , and  $DB-db$ , respectively. Let  $S_0$  be the minimum support specified by the user,  $L$ ,  $L'$ ,  $L''$  be the set of frequent itemsets in  $DB$ ,  $db$ , and  $DB-db$  respectively. Assume that the frequent itemsets  $L$  and the support of each itemset in  $L$  are available in advance. After a period of time, some useless data is deleted from  $DB$ , forming the deleted database  $db$ . Suppose there is an itemset  $X$  in  $DB$ , and we know that  $X$  could be frequent, infrequent or absent in  $db$ . Also suppose the support of  $X$  in  $DB-db$  is  $X_{supp}$ , with respect to the same minimum support  $S_0$ , and  $X$  is expected to be frequent only if  $X_{supp} \geq S_0$ . So, a frequent itemset  $X$  in  $DB$  may no longer be frequent again in  $DB-db$  after some data is deleted from  $DB$ . Similarly, an infrequent itemset  $X$  can become frequent in  $DB-db$ .

There are two maintenance tasks for association rules: i) evaluating the approximate upper and lower support bounds between those itemsets in  $DB$  that are most likely to change their frequentness in the remaining database  $DB-db$ , and ii) finding out all the frequent itemsets in  $DB-db$ .

### 3 A Decremental Algorithm

This section presents our DUA algorithm. In the DUA algorithm, we take into account (1) the changes of frequent and infrequent itemsets in a database when some data is subtracted from the original database; and (2) the approximation range of the itemset support in  $DB$  between those itemsets that have a chance to change their frequentness in  $DB-db$ .

#### Algorithm DUA

**Input:**  $DB$ : the original database with size  $|DB|$ ;  $db$ : the deleted dataset from  $DB$  with size  $|db|$ ;  $L$ : the set of frequent itemsets in  $DB$ ;  $S_0$ : the minimum support specified by the user;

**Output:**  $L''$ : the set of frequent itemsets in  $DB-db$ ;

1. **compute**  $S'$ ; /\*  $S'$  is explained later in this section and also in Section 4.1. \*/
2. **mine**  $db$  with minimum support  $S'$  and **put** the frequent itemsets into  $L'$ ;
3. **for** each itemset in  $L$  **do**
4.     **for** each itemset in  $L'$  **do**
5.         **identify** the frequent itemsets in  $DB-db$ , **eliminate** them from  $L'$  and **store** to  $L''$ ;
6.     randomly **sample** a set of transactions from  $DB$ , denoted as  $db'$ ;
7.     **mine**  $db'$  with the threshold  $S_0$ , and obtain frequent itemsets  $L'''$ ;
8.     **eliminate** the itemsets from  $L'''$  that occur in  $L$  and  $L'$ ;
9. **for** each remaining itemset in  $L'$  and  $L'''$
10.     **scan**  $DB$  to **obtain** their support in  $DB$ ;
11.     **identify** the frequent itemsets in  $DB-db$  and **append** them to  $L''$ ;
12. **end for**;
13. **return**  $L''$ ;
14. **end procedure**;

From the above description, after eliminating the itemsets from  $L'$  and  $L'''$  that are infrequent or frequent in  $DB-db$ , it is obvious that the remaining itemsets in  $L'$  are all infrequent in  $DB$ , while the remaining itemsets in  $L'''$  are infrequent in  $DB$  and they do not occur in  $db$ .

Based on the previous work on the theoretical analysis for determining a lower threshold [Toivonen96], in DUA, we set  $S'$  as  $0.62 \times S_0$ . In our experiments in Section 4, we demonstrate that  $S' = 0.62 \times S_0$  is an appropriate support threshold that satisfies the requirements for both efficiency and accuracy when mining  $db$ .

## 4 Experiments

The above sections have discussed the problem of pattern maintenance when deleting data from a large database, and have proposed an algorithm named DUA to deal with this problem. This section presents experiments we have conducted on a DELL Workstation PWS650 with 2G main memory and 2.6G CPU. The operating system is WINDOWS 2000.

### 4.1 Experiments for Determining $S'$ When Mining $db$

We employ a traditional algorithm *Apriori* [AR94] with a lower minimum support threshold  $S'$  to get some infrequent itemsets in  $db$ . In order to choose an appropriate threshold experimentally, we take into account the theoretical analysis for obtaining a lower threshold [Toivonen96] and have conducted some experiments on synthetic databases.

We first generated a database T10.I4.D100K as  $DB$  and also a  $db$  of 10% transactions randomly deleted from  $DB$ . We set the minimum support  $S_0$  to 1.5% and 1% respectively, and use a threshold of  $S'$  from  $0.3 \times S_0$  to  $0.9 \times S_0$  to mine  $db$ . Those itemsets in  $DB$ , whose supports are greater than the lower bound  $S_{lower}$  (the lower support constraint) and less than  $S_0$ , are most likely to become frequent in  $DB-db$ . We denote these infrequent itemsets in  $db$  as *IIS*. The itemsets whose supports are greater than  $S_0$  and less than  $S_{upper}$  (the upper support constraint) are called *unstable frequent itemsets*, denoted as *UFIS*. Similarly, those itemsets whose supports are less than  $S_0$  and greater than  $S_{lower}$  are called *unstable infrequent itemsets (UIIS)*. Table 1 shows the execution time, total itemsets in  $db$ , *IIS* and *UIIS* when using a different minimum support  $S'$  to mine  $db$ .

It is possible that most of the transactions in  $db$  contain some identical items. For example, the user deleted all the transactions that contain either item "5" or item "8" from the database. Thus, the transactions in  $db$  either contain item "5" or item "8". In Table 2, we use another database T10.I4.D100K as  $DB$ , and construct  $db$  by deleting all transactions that contain item "120" from  $DB$ . The minimum support  $S_0$  is 1.5% and 1% respectively.

As explained before, all the *UIIS* in  $DB$  must be examined in order to find out whether they will become frequent in  $DB-db$ . So a lower threshold  $S'$  is used to find out as many *UIIS* as possible in  $db$ . From these tables, we can see that the computational costs are less with fewer *UIIS* found when  $S'$  is set to  $0.9 \times S_0$ ,  $0.8 \times S_0$  and  $0.7 \times S_0$  respectively. More *UIIS* can be found when  $S'$  is below  $0.6 \times S_0$ , but more computational costs are also required. The appropriate tradeoff point for  $S'$  is in the interval  $[0.6 \times S_0, 0.7 \times S_0]$ . We use  $0.62 \times S_0$  as the minimum support threshold  $S'$  for mining  $db$ .

**Table 1.** Using different  $S'$  for mining db (with random deletions)

$S'$	$S_0$	Execute ime(s)	Total itemsets	IIS	UIIS
$0.9 \times S_0$	1.5%	195.406	266	26	7
	1%	434.265	387	36	13
$0.8 \times S_0$	1.5%	283.891	294	54	13
	1%	525.282	442	91	24
$0.7 \times S_0$	1.5%	341.906	339	99	13
	1%	617.453	482	131	28
$0.6 \times S_0$	1.5%	429.875	387	147	13
	1%	745.625	538	187	28
$0.5 \times S_0$	1.5%	578.937	465	225	13
	1%	887.015	628	277	28
$0.4 \times S_0$	1.5%	746.719	538	298	13
	1%	1090.937	1065	714	28
$0.3 \times S_0$	1.5%	980.485	727	487	13
	1%	1369.718	2658	2307	28

**Table 2.** Using different  $S'$  for mining db (with specified deletions)

$S'$	$S_0$	Execute ime(s)	Total itemsets	IIS	UIIS
$0.9 \times S_0$	1.5%	24.985	1631	52	0
	1%	53.047	1955	106	0
$0.8 \times S_0$	1.5%	32.469	1733	154	0
	1%	77.453	3025	1176	0
$0.7 \times S_0$	1.5%	43.937	1849	270	0
	1%	91.156	1651	1802	1
$0.6 \times S_0$	1.5%	53.047	1955	376	1
	1%	115.25	5397	3818	2
$0.5 \times S_0$	1.5%	75.188	3025	1446	2
	1%	166.546	8867	7018	2
$0.4 \times S_0$	1.5%	115.25	5397	3818	2
	1%	379.578	18103	16254	2
$0.3 \times S_0$	1.5%	380.594	18103	16254	2
	1%	872.64	37301	35452	2

## 4.2 Experiments on Algorithm DUA

We have conducted two sets of experiments to study the performance of DUA. The first set of experiments is done when  $db$  is generated by randomly deleting transactions from  $DB$ . The second set of experiments is done when  $db$  is generated by deleting transactions that contain specified items from  $DB$ .

### 4.2.1 Experiments with $db$ Formed by Random Deletions

We construct  $db$  by randomly deleting some transactions from  $DB$  (T10.I4.D100K), and then use **DUA** on  $db$  to study its performance against the Apriori algorithm. We define the accuracy of **DUA** as the ratio of the number of frequent itemsets found by **DUA** against the number of frequent itemsets found by Apriori in  $DB-db$ .

We set  $|db|=1\%|DB|$ ,  $5\%|DB|$ ,  $10\%|DB|$ ,  $20\%|DB|$  and  $30\%|DB|$  respectively. The following figures present the performance ratio against Apriori and the accuracy of **DUA**.

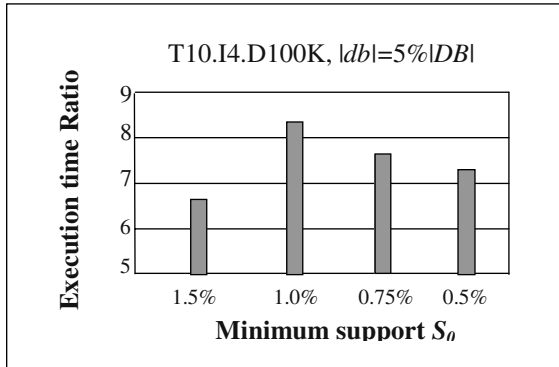


Fig. 1. Performance ratio between DUA and Apriori

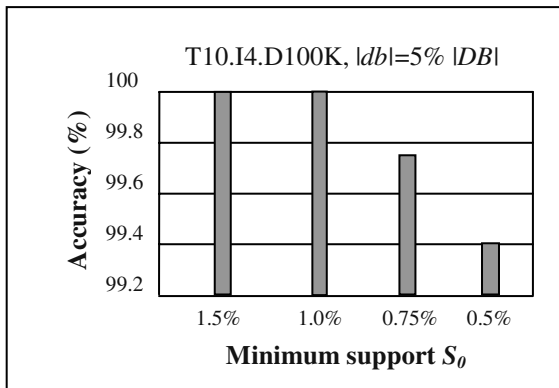


Fig. 2. Accuracy of DUA

Figures 1 and 2 reveal that **DUA** performs 6 to 8 times faster than Apriori for several databases of 100K transactions, and the accuracy of **DUA** is 99.2% to 100%. A trend also can be seen that with the decrease of the minimum support from 1.5% to 0.5%, the accuracy is dropping. The reason is that there are many itemsets with lower supports in a database in general. When the minimum support threshold  $S_0$  is set very low, many itemsets whose supports are slightly below  $S_0$ , namely *UIIS*, may become frequent in *DB-db*. While only a fraction of the *UIIS* can be found in *db* and the sampling database *db'* when using **DUA**. This means that there are some frequent itemsets in *DB-db* that cannot be found using **DUA**. So the accuracy of **DUA** drops when a lower minimum support threshold is given.

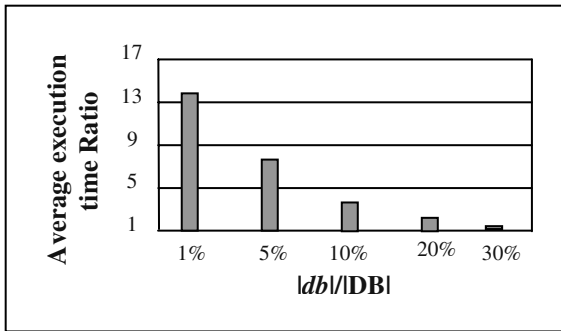


Fig. 3. Execution time against decrement size

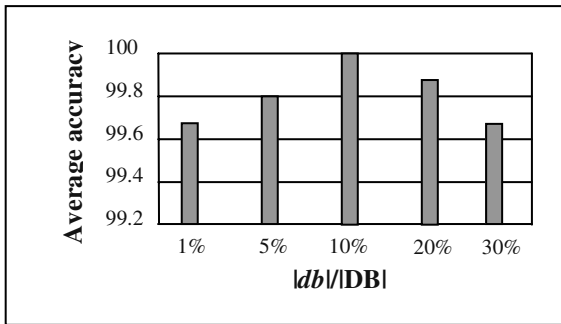


Fig. 4. Average accuracy with different decrement size

With the increase of the size of *db*, the average time ratio against Apriori slows down. In Figures 3 and 4, when  $|db|=30\%|DB|$ , **DUA** is only 1.2 times faster than Apriori. It is clear that Apriori should be applied to mining *DB-db* instead of using **DUA**, when the amount of the data deleted from *DB* is greater than 30% of the total data in *DB*, i.e.,  $|db|>30\%|DB|$ .

#### 4.2.2 Experiments with *db* Formed by Specified Deletions

In the following experiments, we construct *db* by deleting transactions that contain a specified item from *DB*. Figures 5 and 6 present the performance and accuracy of **DUA** on *db* that is formed by deleting transactions containing the specified item “100”, which has a moderate support in *DB*. In this case, DUA is still 6 to 10 times faster than re-running Apriori on *DB-db*.

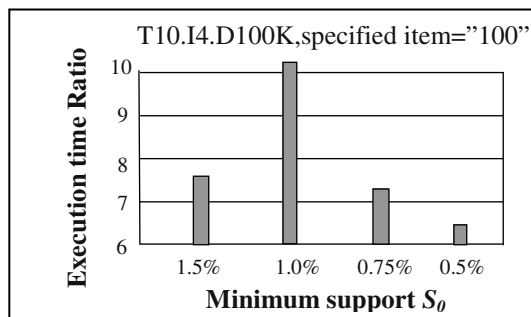


Fig. 5. Performance ratio between DUA and Apriori

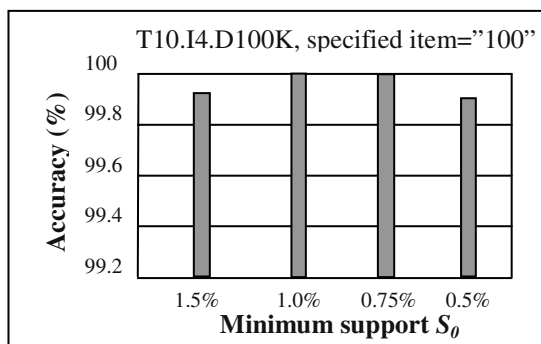


Fig. 6. Accuracy of DUA

## 5 Conclusion and Future Work

Pattern maintenance is a challenging task in data mining and machine learning. In this paper, we have studied the problems of pattern maintenance when data is subtracted from a large database, and have proposed an efficient decremental algorithm to deal with the problem. Our method checks the itemsets found in the subtracted dataset and the itemsets found in the original database in order to determine which itemsets are frequent in the remaining database and which are not frequent any more. Experiments have shown that our **DUA** algorithm is faster than using Apriori on the remaining database, with a high accuracy.

In practice, the operations of data insertions and deletions are interleaving, which make the problem of pattern maintenance in large databases more complicated. Also, a theoretical analysis of the error bounds for **DUA** is very important. These are items for our future work.

## References

- [AR94] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules. *VLDB94*, 1994: 487–499.
- [CHNW96] D. Cheung, J. Han, V. Ng, and C. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. *ICDE'96*, 1996: 106–114.
- [HPY00] J. Han, J. Pei, and Y. Yin, Mining Frequent Patterns without Candidate Generation. *SIGMOD00*, 2000: 1–12.
- [PZOD99] S. Parthasarathy, M. Zaki, M. Ogihara, and S. Dwarkadas, Incremental and Interactive Sequence Mining. *CIKM'99*, 1999: 251–258.
- [TBAR97] S. Thomas, S. Bodagala, K. Alsabti and S. Ranka, An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. *KDD'97*, 1997: 263–266.
- [Toivonen96] H. Toivonen, Sampling Large Databases for Association Rules. *VLDB96*, 1996: 134–145.
- [U94] P. Utgoff, An Improved Algorithm for Incremental Induction of Decision Trees. *ICML'94*, 1994: 318–325.
- [ZZY03] S. Zhang, C. Zhang, and X. Yan, Post-mining: Maintenance of Association Rules by Weighting. *Information Systems*, Vol. 28, 7(2003): 691–707.



# Discovering Richer Temporal Association Rules from Interval-Based Data

Edi Winarko and John F. Roddick

School of Informatics and Engineering,  
Flinders University, PO Box 2100, Adelaide, South Australia 5001,  
{edi.winarko, roddick}@infoeng.flinders.edu.au

**Abstract.** Temporal association rule mining promises the ability to discover time-dependent correlations or patterns between events in large volumes of data. To date, most temporal data mining research has focused on events existing at a point in time rather than over a temporal interval. In comparison to static rules, mining with respect to time points provides semantically richer rules. However, accommodating temporal intervals offers rules that are richer still. In this paper we outline a new algorithm to discover frequent temporal patterns and to generate richer interval-based temporal association rules.

## 1 Introduction

Temporal data mining can be defined as the search for interesting correlations or patterns in large sets of temporal data. Temporal data mining has the capability to discover patterns or rules which might be overlooked when the temporal component is ignored or treated as a simple numeric attribute [1]. A large volume of research has therefore been focused on temporal data mining to discover temporal rules such as sequential patterns [2], episodes [3], temporal association rules [4,5] and inter-transaction association rules [6]. However, almost all of these studies have been focused on data that are stamped with, and interpreted as, time points, whereas intervals, and their relationships, have been largely overlooked. Moreover, the source data is commonly accumulated for other purposes, such as in web server or on line transaction logs, financial data, sensor data or even for backups [7].

Mining temporal rules from interval-based data is more complex and requires a different approach from mining point-based data. An interval has duration and therefore the generated patterns have different semantics than simply *before* and *after*. Allen's temporal interval logic (and its extensions) [8,9,10] are commonly used to describe the relationships among intervals.

There is some previous work on the discovery of temporal patterns from interval-based data [11,12,13]. Villafane *et al.* [11] propose a technique to discover containment relationships from interval time series. While existing techniques consider time series as point-based events, this paper treats time series as interval-based events. One of the applications of containment relationships is

the medical field where containment relationships among diseases can be discovered. For example, we may discover that during a flu infection, a certain strain of skin-borne bacteria is present. However, the containment rules discussed are constrained to the Allen relations *contain* or *during*. Kam and Fu [12] consider the discovery of temporal patterns for interval-based events stored in a temporal database, using an algorithm based on the Apriori algorithm [14]. The algorithm transforms the original database into vertical data format, as used in [15], instead of transforming it into a list of sequences, as in [2].

The problem of discovering temporal patterns and rules from a state sequence is presented in [13]. Suppose  $s$  is a state,  $b$  is the *start-time* of the state, and  $f$  is the *end-time* of a state, a state sequence is defined a series of triples defining state intervals  $(b_1, s_1, f_1), (b_2, s_2, f_2), (b_3, s_3, f_3), \dots, (b_n, s_n, f_n)$ , where  $b_i \leq b_{i+1}$  and  $b_i < f_i$ . A temporal pattern is defined as a set of states together with their interval relationships. These relationships are represented as a square matrix  $R$  whose elements  $R[i, j]$  denote the relationship between state intervals  $i$  and  $j$ . To discover the patterns, the algorithm is based on the Apriori algorithm [14] and is designed to work with a single sequence of states. After all frequent temporal patterns are found, the temporal rule  $X \mapsto Y$  is generated from every pair  $(X, Y)$  of frequent temporal patterns where  $X$  is a subpattern of  $Y$ .

In this paper, we consider the problem of finding temporal patterns in interval-based data. While this is similar to the framework described in [12], our definition of temporal patterns follows the one presented in [13]. We propose a new algorithm by extending the *MEMISP* (**MEM**ory **INDEX**ing for **S**equential **P**attern mining) algorithm [16] for the discovery of the temporal patterns from interval data. We choose to base our algorithm on the *MEMISP* algorithm because the *MEMISP* is more efficient than both *GSP*<sup>1</sup>[17] and *PrefixSpan*<sup>2</sup> [18] algorithms in finding sequential patterns from transactional databases [16]. Furthermore, our database contains a list of interval sequences, which can be viewed as an extension of a list of (point) sequences used in the sequential mining.

Similar to the *MEMISP* algorithm, our algorithm also requires one database scan and does not require candidate generation or database projection. When the database is too large to fit into memory, the algorithm divides the database into several partitions and mines each partition. A second pass of the database is then required to validate the true patterns in the database. After discovering all frequent temporal patterns, we show the method to generate temporal rules from the frequent patterns.

The remainder of the paper is organised as follows. Section 2 discusses the temporal pattern mining problem. The proposed algorithm for mining frequent temporal patterns is explained in section 3. Section 4 defines the temporal rules and describes the method to generate the rules. Our conclusions and areas for further research will be presented in Section 5. Due to space restrictions, some of the discussion is abbreviated and the reader is directed to the longer technical report [19].

---

<sup>1</sup> An Apriori based algorithm.

<sup>2</sup> An FP-Growth based algorithm.

## 2 Problem Statement

**Definition 1.** Given a temporal database  $D = \{t_1 \dots t_n\}$ , each record  $t_i$  consists of a *client-id*, several other attributes, a *start-time*, and an *end-time*, where  $start-time < end-time$ . We assume that the interval between the *start-time* and *end-time*, indicating the interval during which the record values are valid, is a relatively short interval (as compared to the total period under analysis). Each *client-id* can be associated with more than one record. We assume a single temporal attribute is timestamped with the interval, and denote it as a *state*.

In most databases, several temporal attributes can be recorded. Each of these attributes represents a different temporal dimension of the data. For example, in a medical database the date of birth of a patient, the dates of medical examinations, the dates of important medical incidents and other dates concerning different facts of the evolution of the health of a patient can be recorded [20]. In these cases, we can choose one or more temporal attributes as our target in the mining process.

**Definition 2.** Let  $S$  denote the set of all possible states. A state  $s \in S$  that holds during a period of time  $[b, f]$ <sup>3</sup> is denoted as  $(b, s, f)$ , where  $b$  is the *start-time* and  $f$  is the *end-time*. A client's sequence of states is defined as a list of states where each state is associated with the same client such that for client  $i$  we have a state sequence  $cs_i = \langle (b_1, s_1, f_1), (b_2, s_2, f_2), \dots, (b_n, s_n, f_n) \rangle$ , where  $b_i \leq b_{i+1}$  and  $b_i < f_i$ .

**Definition 3.** If  $s$  is a single state type in  $S$ , then  $s$  is a temporal pattern, denoted as  $\langle s \rangle$ .

**Definition 4.** A temporal pattern of size  $n > 1$  is defined by a pair  $(s, R)$ , where  $s : \{1, \dots, n\} \rightarrow S$  maps index  $i$  to the corresponding state, and  $R$  is an  $n \times n$  matrix whose elements  $R[i, j]$  denotes the relationship between interval  $i$  and  $j$ . The number of intervals in the temporal pattern  $P$  is denoted as  $\dim(P)$ . If  $\dim(P) = k$ , then  $P$  is called a  $k$ -pattern.

As for Höppner [13], we use *normalized* temporal patterns in which the state intervals within the patterns are ordered in increasing index according to their start times, end times, and states. Thus, the normalized temporal patterns only require seven relations out of thirteen relations listed in [8], namely, *before* ( $b$ ), *meets* ( $m$ ), *overlaps* ( $o$ ), *is-finished-by* ( $fi$ ), *contains* ( $c$ ), *equals* ( $=$ ), and *starts* ( $s$ ), as shown in Fig. 1. The first five relationships are when the start times differ. In this case, the ordering is based on the start times. If both intervals are identical, we use the order on the states so that we have  $A$  *equals*  $B$ , instead of  $B$  *equals*  $A$ . If the start times are the same and the end times are different, the ordering is based on the end times.

<sup>3</sup> As for most temporal databases, we assume the begin time is inclusive but the end time is not.

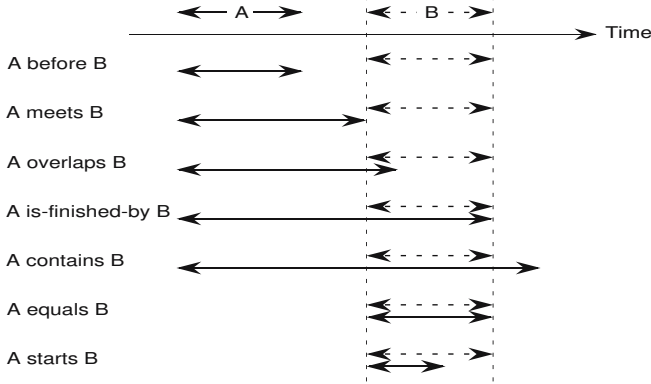


Fig. 1. Seven Relations in Normalized Temporal Patterns

Pat-id	Disease Code	Start Time	End Time	Relative Position
1	A	2	7	
1	E	5	10	
1	B	5	12	
1	D	16	22	
1	C	18	20	
2	D	8	14	
2	C	10	13	
2	G	10	13	
2	F	15	22	
3	A	6	12	
3	B	7	13	
3	D	14	20	
3	C	17	19	
4	B	8	16	
4	A	18	21	
4	D	24	27	
4	E	25	28	

Fig. 2. Database Example

**Definition 5.** The temporal relation  $(s_A, R_A)$  is a subpattern of  $(s_B, R_B)$ , denoted  $(s_A, R_A) \sqsubseteq (s_B, R_B)$ , if  $\dim(s_A, R_A) \leq \dim(s_B, R_B)$  and there is an injective mapping  $\pi : \{1, \dots, \dim(s_A, R_A)\} \rightarrow \{1, \dots, \dim(s_B, R_B)\}$  such that  $\forall i, j \in \{1, \dots, \dim(s_A, R_A)\}: s_A(i) = s_B(\pi(i)) \wedge R_A[i, j] = R_B[\pi(i), \pi(j)]$

**Definition 6.** A client state sequence  $C$  supports a pattern  $P = (s_P, R_P)$  if  $(s_P, R_P) \sqsubseteq (s_C, R_C)$ , where  $(s_C, R_C)$  is a pattern that represent the relationships between intervals in the client state sequence. The support of a pattern  $P$  is defined as  $\sigma(P) = \frac{|D_P|}{|D|}$ , where  $|D_P|$  is the number of client sequences that support the pattern  $P$ , and  $|D|$  is the number of clients in the database  $D$ .

**Definition 7.** Given a minimum support  $minsup$ , a pattern is called *frequent* if its support is greater than or equal to  $minsup$ .

**Table 1.** Frequent Temporal Patterns

1-patterns	$(A) (\sigma = 75\%), (B) (\sigma = 75\%), (C) (\sigma = 75\%),$ $(D) (\sigma = 100\%), (E) (\sigma = 50\%)$
2-patterns	$\begin{pmatrix} A & B \\ = & o \\ * & = \end{pmatrix} (\sigma = 50\%), \begin{pmatrix} A & D \\ = & b \\ * & = \end{pmatrix} (\sigma = 75\%), \begin{pmatrix} A & C \\ = & b \\ * & = \end{pmatrix} (\sigma = 50\%),$ $\begin{pmatrix} B & D \\ = & b \\ * & = \end{pmatrix} (\sigma = 75\%), \begin{pmatrix} B & C \\ = & b \\ * & = \end{pmatrix} (\sigma = 50\%), \begin{pmatrix} D & C \\ = & c \\ * & = \end{pmatrix} (\sigma = 75\%)$
3-patterns	$\begin{pmatrix} A & B & D \\ = & o & b \\ * & = & b \\ * & * & = \end{pmatrix} (\sigma = 50\%), \begin{pmatrix} A & B & C \\ = & o & b \\ * & = & b \\ * & * & = \end{pmatrix} (\sigma = 50\%),$ $\begin{pmatrix} B & D & C \\ = & b & b \\ * & = & c \\ * & * & = \end{pmatrix} (\sigma = 50\%), \begin{pmatrix} A & D & C \\ = & b & b \\ * & = & c \\ * & * & = \end{pmatrix} (\sigma = 50\%)$
4-patterns	$\begin{pmatrix} A & B & D & C \\ = & o & b & b \\ * & = & b & b \\ * & * & = & c \\ * & * & * & = \end{pmatrix} (\sigma = 50\%)$

Given a temporal database and a *minsup*, our goal is to discover all frequent patterns whose support is greater than or equal to *minsup*.

As an example, suppose we are given a temporal database *D*, which stores a list of clinical records, as shown in Fig. 2. Each record contains a *patient-id*, a *disease code* and a pair of ordered time points, indicating the period during which the patient exhibited a given disease. Records in the database have been sorted on the *patient-id*, the *start* time, the *end* time, and the *disease code*. The last column in the table is used to visualize the relative position of state intervals in each patient. Using a *minsup* of 40%, the frequent temporal patterns are shown in Table 1<sup>4</sup>.

### 3 Discovery of Temporal Patterns

Our proposed algorithm discovers temporal patterns in three steps. First, the algorithm reads the database into memory. While reading the database, it counts the support of each state and generates frequent 1-patterns. The algorithm then constructs an index set for each frequent 1-pattern and finds frequent patterns using the state sequences indicated by elements of the index set. Finally, using a recursive *find-then-index* strategy, the algorithm discovers all temporal patterns from the in-memory database. Each of these steps is described in the following sections. Pseudo code of our algorithm is shown in Algorithm 1. To illustrate it, we use the algorithm to discover frequent patterns from the example database shown in Fig. 2 and a *minsup* of 40%. Definition 8 is needed for describing the algorithm.

<sup>4</sup> For brevity, we do not put labels on the rows of the matrix because they are always similar to the column labels.

**Definition 8.** Given a pattern  $\rho$ , where  $\dim(\rho) = n$ , and a frequent 1-pattern  $\langle s \rangle$  in the database, a pattern  $\rho'$  of size  $(n+1)$  can be formed by adding the  $s$  as a new element to  $\rho$  and setting the relationships between  $s$  and each element of  $\rho$ . The frequent 1-pattern  $\langle s \rangle$  is called *stem-state* of the pattern  $\rho'$  and  $\rho$  is the *Prefix pattern* (abbreviated as *P-pat*) of  $\rho'$ .

**Input:** : a temporal database  $D$ ,  $\text{minsup}$   
**Output:** : all frequent normalized temporal patterns

- 1: **while** scan  $D$  into in-memory database  $MDB$  **do**
- 2:   find  $S =$  the set of all frequent 1-patterns
- 3: **end while**
- 4: **for** each  $s \in S$  **do**
- 5:   form the pattern  $\rho = \langle s \rangle$  and output  $\rho$ ;
- 6:   call  $\text{IndexSet}(s, \langle \rangle, MDB)$  ;
- 7:   call  $\text{Mine}(\rho, \rho\text{-idx})$  ;
- 8: **end for**

**Algorithm 1.** Algorithm to Generate Frequent Temporal Patterns

### Step 1 - Reading the Database into Memory

In this first step, the algorithm reads the database  $D$  into memory. The in-memory database is referred to as  $MDB$ . While reading each client sequence from the database, the algorithm computes the support count of every state, then finds the set of all frequent 1-patterns  $S$ . From the example database, the algorithm finds frequent 1-patterns  $\langle A \rangle$  ( $\sigma = 75\%$ , supported by 3 client sequences  $cs_1$ ,  $cs_3$ , and  $cs_4$ ),  $\langle B \rangle$  ( $\sigma = 75\%$ ),  $\langle C \rangle$  ( $\sigma = 75\%$ ),  $\langle D \rangle$  ( $\sigma = 100\%$ ), and  $\langle E \rangle$  ( $\sigma = 50\%$ ). Therefore, we have a set of frequent 1-patterns  $S = \{A, B, C, D, E\}$ .

### Step 2 - Constructing the Index Set

In this step, the algorithm first outputs the pattern  $\rho$  formed by combining current *P-pat* and a stem  $s \in S$ , and then constructs the index set  $\rho\text{-idx}$ . Initially, the *P-pat* is an empty pattern  $\langle \rangle$ . Each element of the index set contains three fields **ptr\_cs**, **a\_intv**, and **pos**, where **ptr\_cs** is a pointer to a client sequence  $cs$ , **a\_intv** is an array of intervals in a client sequence  $cs$  which become part of a pattern  $\rho$ , and **pos** is the first occurring position of  $s$  in a customer sequence  $cs$ . An index element  $(\text{ptr\_cs}, \text{a\_intv}, \text{pos})$  for a client sequence  $cs$  is created only if  $cs$  supports  $\langle s \rangle$ . Subroutine 2 shows the subroutine for constructing an index set.

Continuing on our example, we start with a stem  $s = A$  and the *P-pat* =  $\langle \rangle$ . We output a pattern  $\rho = \langle A \rangle$ , and then call  $\text{IndexSet}(A, \langle \rangle, MDB)$  to construct the index set  $\langle A \rangle\text{-idx}$ . This index set is shown in Fig. 3(a), in which **pos** is 1 for  $cs_1$  and  $cs_3$ , and 2 for  $cs_4$ . The array **a\_intv** contains an interval of a state within a pattern  $\rho = \langle A \rangle$ , i.e., a state  $A$ . Note that  $cs_2$  is not pointed to by any pointer in the index set because it does not support  $\langle A \rangle$ .

```

1: // Construct the index set  $\rho'$ -idx
2: //  $\rho'$  is a pattern formed by combining  $\rho$  and  $s$ 
3: Subroutine IndexSet( $s, \rho, \text{range-set}$ )
4: for each client sequence  $cs$  in  $\text{range-set}$  do
5:   if  $\text{range-set} = \text{MDB}$  then
6:      $\text{start-pos} = 0$ 
7:   else
8:      $\text{start-pos} = \text{pos}$ 
9:   end if
10:  for  $\text{pos} = (\text{start-pos}+1)$  to  $|cs|$  do
11:    if stem state  $s$  is first found at position  $\text{pos}$  in  $cs$  then
12:      insert  $(\text{ptr\_cs}, a\_intv, \text{pos})$  to the index set  $\rho'$ -idx, where  $\text{ptr\_cs}$  points to  $cs$ 
13:    end if
14:  end for
15: end for
16: return index set  $\rho'$ -idx

```

**Subroutine 2.** Subroutine to Construct an Index Set

### Step 3 - Mining Patterns from the Index Set

In this step, the algorithm uses the index set  $\rho$ -idx to find stems with respect to  $P$ -pat  $\rho$ . Any state appearing after the  $\text{pos}$  position in the client sequence  $cs$  pointed by  $\text{ptr\_cs}$  of the entry  $(\text{ptr\_cs}, a\_intv, \text{pos})$  in  $\rho$ -idx could be a potential stem (with respect to  $\rho$ ). Thus, for every  $cs$  existing in  $\rho$ -idx, the algorithm increases the support count of such state by one. The algorithm then determines the set of all stems  $S$  having enough support to form patterns. A subroutine for mining an index set is shown in Subroutine 3.

```

1: // Mine patterns from an index set  $\rho$ -idx;
2: Subroutine Mine( $\rho, \rho$ -idx);
3: for each  $cs$  pointed by  $\text{ptr\_cs}$  of an entry  $(\text{ptr\_cs}, a\_intv, \text{pos})$  in  $\rho$ -idx do
4:   for  $\text{pos} = \text{pos} + 1$  to  $|cs|$  in  $cs$  do
5:      $\text{count}(s) = \text{count}(s) + 1$ , where  $s$  is a potential stem state
6:   end for
7: end for
8: find  $S =$  the set of stems  $s$  having enough support to form a pattern;
9: for each stem state  $s \in S$  do
10:  output the pattern  $\rho'$  by combining  $P$ -pat  $\rho$  and stem  $s$ ;
11:  call IndexSet( $s, \rho, \rho$ -idx) // to construct the index set  $\rho'$ -idx;
12:  call Mine( $\rho', \rho'$ -idx) // to mine patterns with index set  $\rho'$ -idx;
13: end for

```

**Subroutine 3.** Subroutine to Mine an Index Set

We continue our example by processing  $\langle A \rangle$ -idx (Fig. 3(a)). This can be done by calling *Mine*( $\langle A \rangle, \langle A \rangle$ -idx). We process each client sequence  $cs$  pointed by a

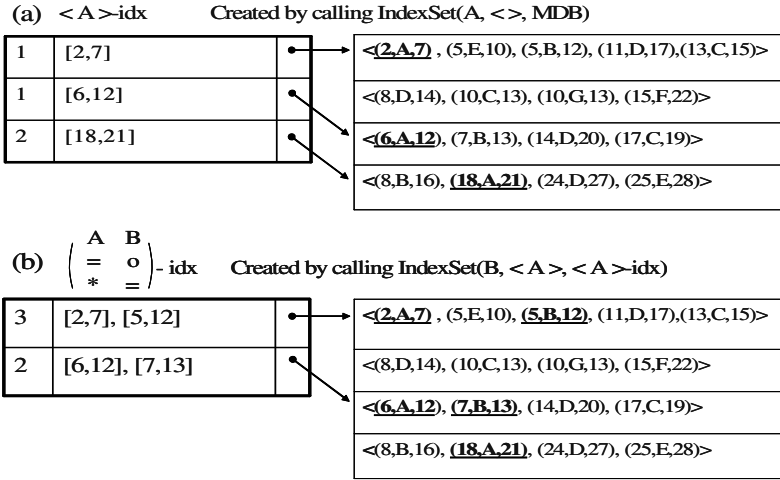


Fig. 3. Examples of Index Sets

pointer in  $\langle A \rangle$ -idx. Since the *pos* of the  $(ptr\_cs, a\_intv, pos)$  pointing to  $cs_1$  is 1, we only count the states occurring after position 1. Therefore, the support count

of a potential stem  $E$  for a potential pattern  $P2_E = \begin{pmatrix} A & E \\ = & o \\ * & = \end{pmatrix}$  is increased

by one. There is also a stem  $B$  for a pattern  $P2_B = \begin{pmatrix} A & B \\ = & o \\ * & = \end{pmatrix}$ , a stem  $D$  for

a pattern  $P2_D = \begin{pmatrix} A & D \\ = & b \\ * & = \end{pmatrix}$ , and a stem  $C$  for a pattern  $P2_C = \begin{pmatrix} A & C \\ = & b \\ * & = \end{pmatrix}$ .

Using the same process, we count states occurring after position 1 and 2 for client sequences  $cs_3$  and  $cs_4$ , respectively. After validating the support counts, we obtain stems  $B$  ( $\sigma = 50\%$ ),  $C$  ( $\sigma = 50\%$ ), and  $D$  ( $\sigma = 75\%$ ) to form patterns  $P2_B$ ,  $P2_C$ , and  $P2_D$ , respectively.

Next, we call  $IndexSet(s, \rho, \rho$ -idx) and  $Mine(\rho', \rho'$ -idx) recursively, where  $s \in \{B, C, D\}$ ,  $\rho = \langle A \rangle$ , and  $\rho'$  is a pattern formed by combining  $\rho$  and  $s$ . We proceed with the mining by taking  $P$ -pat  $\rho = \langle A \rangle$  and  $s = B$ . Combining  $\rho =$

$\langle A \rangle$  and  $s = B$ , we obtain a pattern  $\rho' = \begin{pmatrix} A & B \\ = & o \\ * & = \end{pmatrix}$ . We then call  $IndexSet(B,$

$\langle A \rangle, \langle A \rangle$ -idx) to construct the index set  $\rho'$ -idx. To create  $\rho'$ -idx, we only need to check the client sequences indicated by the index set  $\langle A \rangle$ -idx, rather than in  $MDB$ . Therefore, the search for the occurring position of a stem  $B$  (with respect to  $P$ -pat  $= \langle A \rangle$ ) finds a state  $B$  at position 3 in  $cs_1$  and 2 in  $cs_3$ . The interval values of a state  $B$  in  $cs_1$  and  $cs_3$  are added to the array  $a\_intv$ . No entry is created for  $cs_4$  since a state  $B$  that can produce a pattern  $\rho'$  cannot be found



after position 2. We thus have a new index set  $\rho'$ -idx as shown in Fig. 3(b). The index set  $\langle A \rangle$ -idx is stored for later mining.

Calling  $Mine(\rho', \rho'$ -idx), we find stems  $C$  ( $\sigma = 50\%$ ), and  $D$  ( $\sigma = 50\%$ ) to

$$\text{form patterns } P3_C = \begin{matrix} & A & B & C \\ (= & o & b \\ * & = & b \\ * & * & = \end{matrix} \text{ and } P3_D = \begin{matrix} & A & B & D \\ (= & o & b \\ * & = & b \\ * & * & = \\ & A & B \end{matrix}, \text{ respectively.}$$

Continuing the recursive process on  $P$ -pat  $\rho = \begin{pmatrix} (= & o \\ * & = \end{pmatrix}$  and a stem  $C$ , we output a pattern  $\rho' = P3_C$ . Since there are no further patterns we stop this

process. Next, we continue with  $P$ -pat  $\rho = \begin{pmatrix} (= & o \\ * & = \end{pmatrix}$  and a stem  $D$ . We output a pattern  $\rho' = P3_D$ , then create and mine  $\rho'$ -idx to find a stem  $C$ .

Taking  $P$ -pat  $\rho = P3_D$  and a stem  $C$ , we output a pattern  $\rho'$  and create

$$\text{an index set } \rho'$$
-idx, where  $\rho' = \begin{matrix} & A & B & D & C \\ (= & o & b & b \\ * & = & b & b \\ * & * & = & c \\ * & * & * & = \end{matrix}$ . The mining of  $\rho'$ -idx finds

no more stems. Since we cannot continue the recursive process, we repeat the process by taking  $P$ -pat  $\rho = \langle A \rangle$  and a stem  $s$ , where  $s \in \{C, D\}$ . This process

$$\text{will generate patterns: } \begin{matrix} & A & C \\ (= & b \\ * & = \end{matrix}, \begin{matrix} & A & D \\ (= & b \\ * & = \end{matrix}, \text{ and } \begin{matrix} & A & D & C \\ (= & b & b \\ * & = & c \\ * & * & = \end{matrix}. \text{ At this stage,}$$

we have finished the mining process of a stem  $A$  with  $P$ -pat  $\rho = \langle \rangle$ . All frequent patterns can be discovered by continuing the mining process on stems  $B, C, D$ , and  $E$  with  $P$ -pat  $\rho = \langle \rangle$ .

### 4 Generating Temporal Rules

After all frequent temporal patterns have been discovered, we can generate temporal rules based on Definition 9 below, which was first introduced in [13].

**Definition 9.** A temporal rule  $X \Rightarrow Y$  can be constructed from every pair  $(X, Y)$  of frequent temporal patterns with  $X \sqsubseteq Y$  ( $X$  is a subpattern of  $Y$ ). The confidence of a temporal rule  $conf(X \Rightarrow Y) = \frac{\sigma(Y)}{\sigma(X)}$ . Given a minimum confidence  $minconf$ , the temporal rule  $X \Rightarrow Y$  will be generated if its confidence is greater than or equal to  $minconf$ .

We are interested in generating the *forward* rules, that is, the rules that are used for predicting the future rather than in the past. As an example, from a

$$\text{frequent pattern } X = \begin{matrix} & A & B \\ (= & o \\ * & = \end{matrix} \text{ and a temporal pattern } Y = \begin{matrix} & A & B & D \\ (= & o & b \\ * & = & b \\ * & * & = \end{matrix}$$

(see table 1), we can get a rule  $X \Rightarrow Y$  with the confidence of 100%. This rule means that *if A overlaps B occurs, then it is highly likely that A before D and B before D will also occur.*

The rule generation process works as follows. If  $Y$  is a frequent  $n$ -pattern, where  $n > 1$ , and  $S = \langle s_1, s_2, \dots, s_n \rangle$  is a list of states in  $Y$  ( $S$  has been ordered in increasing index according to the order within  $Y$ ), then we will find all subpatterns  $X$  of  $Y$  which have ordered list of states  $S_i = \langle s_1, s_2, \dots, s_i \rangle$ ,  $i = 1, 2, \dots, (n - 1)$ . Therefore, for each frequent  $n$ -pattern  $Y$ , there are at most  $(n - 1)$  subpatterns  $X$  of  $Y$ . For every such subpattern  $X$ , we generate a rule of the form  $X \Rightarrow Y$  if its confidence is greater than or equal to *minconf*.

The process of finding  $X$  starts by taking  $i = (n - 1)$  so that  $X$  has a list of states  $S_{n-1} = \langle s_1, s_2, \dots, s_{n-1} \rangle$ . If  $X \Rightarrow Y$  has enough confidence, we will generate this rule and continue to test the next subpattern. If  $X \Rightarrow Y$  does not have enough confidence, we do not have to check for  $X$  where  $i < (n - 1)$  because the generated rule will have a smaller confidence.

## 5 Conclusion and Future Work

In this paper we have studied the discovery of temporal patterns along with their temporal association rules from interval-based data. We have proposed a new algorithm by extending *MEMISP*, an existing algorithm for mining sequential patterns, to discover the frequent temporal patterns from interval-based data. The proposed algorithm is illustrated using an example. Our implementation of the algorithm which incorporates *maximum gap* time constraint and the results of our experiments are presented in [19]. Future work includes an application to real-world problem domains and enhancements to the interface to facilitate the discovery of temporal patterns and rules directly from relational temporal databases.

## References

1. Roddick, J.F., Spiliopoulou, M.: A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering* **14** (2002) 750–767
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: *Proceedings of 11th International Conference on Data Engineering*. (1995) 3–14
3. Mannila, H., Toivonen, H., Verkamo, A.: Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* **1** (1997) 259 – 289
4. Li, Y., Ning, P., Wang, X.S., Jajodia, S.: Discovering calendar-based temporal association rules. In: *Proceedings of the 8th International Symposium on Temporal Representation and Reasoning*. (2001) 111–118
5. Ozden, B., Ramaswamy, S., Silberschatz, A.: Cyclic association rules. In: *Proceedings of the 14th International Conference on Data Engineering*. (1998) 412–421
6. Lu, H., Feng, L., Han, J.: Beyond intratransaction association analysis: mining multidimensional intertransaction association rules. *ACM Transactions on Information Systems* **18** (2000) 423 – 454

7. Bettini, C., Wang, X.S., Jajodia, S.: Testing complex temporal relationships involving multiple granularities and its application to data mining. In: Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Montreal, Canada (1996) 68–78
8. Allen, J.: Maintaining knowledge about temporal intervals. *Communications of the ACM* **26** (1983) 832–843
9. Freksa, C.: Temporal reasoning based on semi-intervals. *Artificial Intelligence* **54** (1992) 199–227
10. Roddick, J.F., Mooney, C.H.: Linear temporal sequences and their interpretation using midpoint relationships. *IEEE Transactions on Knowledge and Data Engineering* **17** (2005) 133–135
11. Villafane, R., Hua, K.A., Tran, D., Maulik, B.: Mining interval time series. In: *Data Warehousing and Knowledge Discovery*. (1999) 318–330
12. Kam, P.S., Fu, A.W.C.: Discovering temporal patterns for interval-based events. In: *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000)*, London, UK (2000) 317–326
13. Höppner, F.: Learning temporal rules from state sequence. In: *Proceedings of IJ-CAI Workshop on Learning from Temporal and Spatial Data*, Seattle, USA (2001) 25–31
14. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. (1994) 487–499
15. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. *Machine Learning Journal* **42** (2001) 31–60
16. Lin, M.Y., Lee, S.Y.: Fast discovery of sequential patterns by memory indexing. In: *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2002)*, Aix-en-Provence, France (2002) 150–160
17. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: *Proceedings of the 5th International Conference on Extending Database Technology*, Avignon, France (1996) 3–17
18. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Prefixspan: Mining sequential patterns efficiently by prefixprojected pattern growth. In: *Proceedings of the 2001 International Conference on Data Engineering (ICDE'01)*, Heidelberg, Germany (2001) 215–224
19. Winarko, E., Roddick, J.F.: Discovering richer temporal association rules from interval-based data : Extended report. Technical SIE-05-003, School of Informatics and Engineering, Flinders University (2005)
20. Koundourakis, G., Theodoulidis, B.: Association rules and evolution in time. In: *Proceedings of Methods and Applications of Artificial Intelligence, Second Hellenic Conference on AI, SETN 2002*, Thessaloniki, Greece (2002) 261–272

# Semantic Query Expansion Combining Association Rules with Ontologies and Information Retrieval Techniques

Min Song, Il-Yeol Song, Xiaohua Hu, and Robert Allen

College of Information Science & Technology,  
Drexel University, Philadelphia, PA 19104,  
(215) 895-2474, 01  
{min.song, song, thu, bob.allen}@drexel.edu

**Abstract.** Query expansion techniques are used to find the desired set of query terms to improve retrieval performance. One of the limitations with the query expansion techniques is that a query is often expanded only by the linguistic features of terms. This paper presents a novel semantic query expansion technique that combines association rules with ontologies and information retrieval techniques. We propose to use the association rule discovery to find good candidate terms to improve the retrieval performance. These candidate terms are automatically derived from collections and added to the original query. Our method is differentiated from others in that 1) it utilizes the semantics as well as linguistic properties of unstructured text corpus and 2) it makes use of contextual properties of important terms discovered by association rules. Experiments conducted on a subset of TREC collections give quite encouraging results. We achieve from 15.49% to 20.98% improvement in term of P@20 with TREC5 ad hoc queries.

## 1 Introduction

A typical goal of an information retrieval (IR) system is to find a set of documents containing information needed by searchers in the given indexed database(s). In processing queries that searchers formulate, the conventional IR query languages require the searcher to state precisely what they want. Searchers need to be able to express their needs in terms of precise queries (either in Boolean form or natural languages). However, due to searchers' lack of knowledge in the search domain (anomalous state of knowledge -- An anomaly in one's state of knowledge, or lack of knowledge, with respect to a problem faced), a query syntax formulated by searchers often does not meet the searchers' information needs. In addition, a single term based query that a normal user formulates retrieves many irrelevant articles as well as fails to find hidden knowledge or relationships buried in content of the articles.

To overcome this issue with query formulation, many IR systems provide facilities for relevance feedback, with which searchers can identify documents of interest to them. IR systems can then use the keywords assigned to these desired documents to find other potentially relevant documents. However, these IR systems fail to distinguish among the attributes of the desired documents for their relative importance to the searchers' needs.

With these issues in current Query Expansion (QE) techniques in mind, we introduce a novel querying technique, called *SemanQE*, combining association rules and ontologies and IR techniques to retrieve promising documents for information extraction. *SemanQE* has several unique strengths over other QE techniques. First, it proposes a hybrid query expansion algorithm combining association rules with ontologies and natural language processing techniques. Second, our technique utilizes the semantic as well as linguistic properties of unstructured text corpus. Third, our technique makes use of contextual properties of important terms discovered by association rules. To evaluate the performance of *SemanQE*, cosine similarity-based QE and *SLIPPER*, a rule-based QE technique are compared. We also investigate whether ontologies impact on the retrieval performance.

This paper makes the following contributions: (1) our method utilizes the semantics as well as linguistic properties of unstructured text corpus and thus our system is able to expand queries based on indirect associations embedded among the terms (2) Our method makes use of contextual properties of important terms discovered by association rules. (3) We demonstrate the effectiveness of our method through experiments. The experiments conducted on a subset of TREC collections give quite encouraging results. We achieve from 15.49% to 20.98% improvement in term of P@20 with TREC5 ad hoc queries.

The rest of paper consists of the following chapters: Section 2 summarizes the related work. Section 3 describes the overall architecture of *SemanQE*. Section 4 describes the evaluation. Section 5 concludes the paper.

## 2 Related Works

The quality of a query fed to an IR system has a direct impact on the success of the search outcome. In fact, one of the most important but frustrating tasks in IR is query formulation [3]. Relevance feedback is a popular and widely accepted query reformulation strategy. The main idea consists of selecting important terms, or expressions, attached to the documents that have been identified as relevant by the user, and of enhancing the importance of these terms in a new query formulation. The expected effect is that the new query will be moved towards the relevant documents and away from the non-relevant ones.

Pseudo-relevance feedback methods improve retrieval performance on average but the results are not as good as relevance feedback. In pseudo-relevance feedback, problems arise when terms or phrases taken from assumed-to-be relevant documents that are actually non-relevant are added to the query causing a drift in the focus of the query. To tackle this issue, Mitra, et al. [9] incorporated term co-occurrences to estimate word correlation for refining the set of documents used in query expansion.

Mihalcea and Moldovan [8] found that using the selected passages from documents for query expansion is effective in reducing the number of inappropriate feedback terms taken from non-relevant documents. Lam-Adesina and Jones [5] applied document summarization to query expansion. In their approach, only terms present in the summarized documents are considered for query expansion. Lam-Adesina and Jones adopted a summarization technique based on sentence-extracted summaries that are found by scoring the sentences in the documents. The scoring method is simply a sum of the scores gained by the four summarization methods: 1) Luhn's keyword

cluster, 2) title terms frequency, 3) location/header, and 4) query-bias methods. Whereas their technique is based on simple mathematical properties of terms, our techniques are information theory-based as well as mathematically solid.

Liu et al. [6] used noun phrases for query expansion. Specifically, four types of noun phrases were identified: proper names, dictionary phrases, simple phrases, and complex phrases. A document has a phrase if all the content words are in the phrase within the defined window, and these documents that have matched phrases are considered to be relevant. They also apply a similarity measure to select the content words in the phrases to be positively correlated in the collection.

Latiri et al. [7] approached query expansion by considering the term-document relation as fuzzy binary relations. Their approach to extract fuzzy association rules is based on the closure of an extended fuzzy Galois connection, using different semantics of term membership degrees.

Because we also investigate whether adding concepts from WordNet to query sets by SemanQE improves the retrieval performance, we briefly survey some related works to our approach. Liu et al. [6] add selected synonyms, hyponyms, and compound words based on their word sense disambiguation technique. Our approach to word sense disambiguation is different in that we disambiguate word sense by similarity criteria between all the non-stopwords from the synonyms and definitions of the hyponym synsets and keyphrases extracted from the retrieved documents. Voorhees' [12] used WordNet for adding synonyms of query terms whereas we use WordNet to add synonyms and substantial hyponyms of the top N ranked terms and phrases.

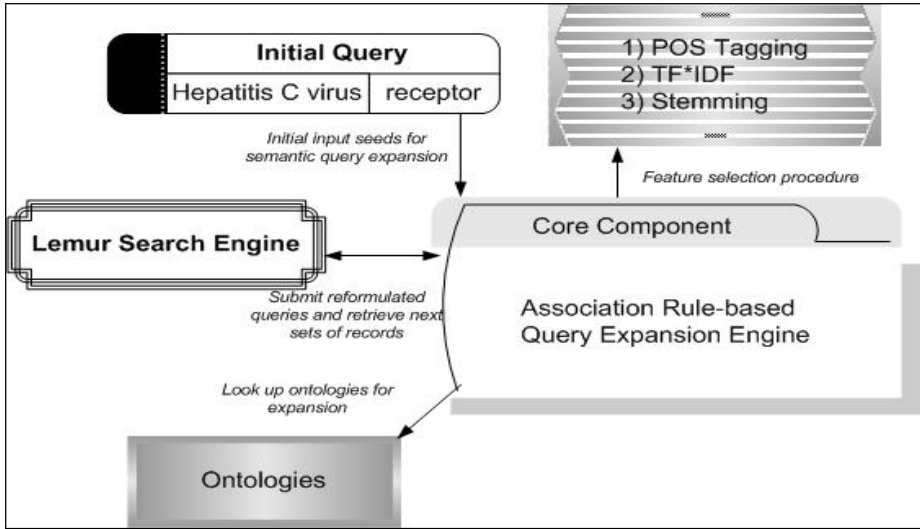
### 3 The Semantic Query Expansion System

In this section, we describe the semantic query expansion system. In Section 3.1, we present the system architecture of our semantic query expansion system. In Section 3.2, we discuss the ontology used in our method. Finally, Section 3 explains our semantic query expansion algorithm called SematicQE.

#### 3.1 The System Architecture

The system architecture of our semantic query expansion system, SemantiocQE, is illustrated in Fig. 1. SemanQE consists of three major components: 1) core association rule-based query expansion 2) feature selection, and 3) ontologies-based expansion components.

We use the Lemur IR system as a backend engine for SemanQE. Lemur is developed by collaboration between the Computer Science Department at the University of Massachusetts and the School of Computer Science at Carnegie Mellon University. Lemur is designed to facilitate research in language modeling and information retrieval. The core association rule-based expansion algorithm is based on a well-known Apriori algorithm [1]. The apriori algorithm has been widely used to mine useful knowledge in large transaction databases. The support of a set of items in a transaction database is the fraction of all transactions containing the itemset. An itemset is called frequent if its support is greater or equal to a user-specified support threshold. An association rule is an implication of the form  $X \Rightarrow Y$  where  $X$  and  $Y$



**Fig. 1.** System architecture of SemanQE

are disjoint itemsets. To apply association rule mining to our query expansion, we assume that each document can be seen as a transaction while each separate word inside can also be seen as items, represented by wordset.

The feature selection component processes the input documents to select important terms. In doing so, unimportant words such as functional words and stop words are excluded. We applied TF\*IDF technique to extract important terms and phrases. In addition, we applied a POS tagging technique to filter out less important terms in terms of POS tags. TF\*IDF was first proposed by Salton and Buckley [10]. It is a measure of importance of term in a document or class. Brill POS Tagger is chosen for our POS tagger. Brill's technique is one of the high quality POS tagging techniques.

Ontologies component expands queries selected from the core component. WordNet is used as ontologies for our system. With a set of terms and phrases, WordNet is referenced to find relevant entries semantically and syntactically.

The outline of the approach described in Figure 1 is as follows:

**Step 1:** Starting with a set of user-provided seed instances (the seed instance can be quite small), our system retrieves a sample of documents from the backend indexes via a search engine. At the initial stage of the overall document retrieval process, we have no information about the documents that might be useful for extraction. The only information we require about the target answer sets is a set of user-provided seed instances. We use some simple queries (just use the attribute values of the initial seed instances) to extract the document sample of pre-defined size from the search engine.

Step 2: On the retrieved document set, we parse each document into sentences and apply IR and natural language processing techniques to select important terms and phrases from the input documents.

Step 3: Applying a hybrid querying expansion algorithm that combines association rules and ontologies to derive queries targeted to match and retrieve additional documents similar to the positive examples.

Step 4: Reformulate queries based on the results of Step 3 and query the search engine again to retrieve the improved result sets matched to the initial queries.

Fig. 2 shows the how SemanQE works and what output it generates in each step.

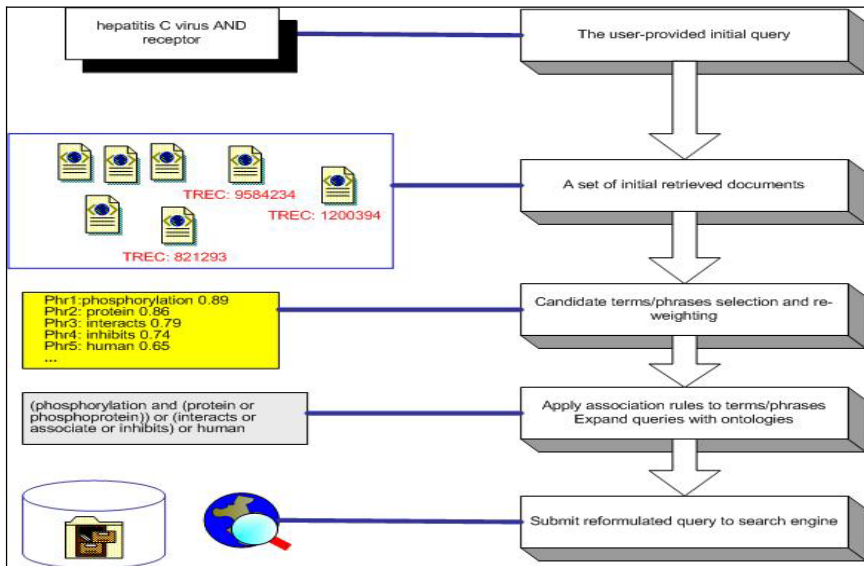


Fig. 2. Procedures of the System Workflow

### 3.2 Ontologies

We adopted WordNet for ontologies of SemanQE. WordNet is an online lexical reference system in which English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. WordNet was developed in the Cognitive Science Laboratory at Princeton University. Published results on sense based query expansions are not very recent [11][12]. A more recent work [4] analyzes the effect of expanding a query with WordNet synsets, in a "canned" experiment where all words are manually disambiguated. Our usage of WordNet is for retrieving the promising documents by expanding queries syntactically and semantically. We traverse WordNet hierarchy to find out the best WordNet entries for the given terms to be expanded.



### 3.3 The SemanQE Algorithm

In this section, we provide details of SemanQE algorithm. As shown in Table 1,

**Table 1.** Association Rule-based SemanQE Algorithm

---

(1) Retrieve initial results from Lemur based on the queries provided by a user
(2) Select important noun and phrases from CL by POS and TF*IDF
(3) Apply Apriori to find all X->Y rules
For ( $i = 1; i < \text{Size of } L_i; i++$ ) do
(4) Build $Q_i$ based on rules generated by Step 2.
(5) Apply Ontologies to expand $Q_i$ .
(6) Query Lemur with $Q_i$ constructed by Step 5
If hit count $\neq 0$
(7) Retrieve TREC records for information extraction

---

SemanQE takes the user-provided queries to retrieve the initial set of documents from Lemur. The general description of the algorithm is as follows: Once the data was parsed, the important noun and noun phrases were extracted based on the following two techniques: TF\*IDF and Brill's Part of Speech (POS) tagging technique [2]. TF\*IDF stands for Term Frequency \* Inverse Document Frequency, which is a powerful IR technique to select important terms. We make use of Brill's POS tagging technique to capture candidate noun and noun phrases. After the important noun and noun phrases are extracted, Apriori algorithm [1] is applied. SemanQE builds a set of queries based on the rules generated by Apriori. We finally applied ontologies to expand queries generated by association rules. The example of the query is as follows:

**(Adult+AND+Antineoplastic+Combined+Chemotherapy+Protocols+AND+Dacarbazine)+NOT+raynaud**

Lemur was then searched with the query constructed by Step 4 and retrieve TREC records. In the feature selection, important terms or phrases are represented in the following term x document matrix.

$$D_i = t_{i1}, t_{i2}, \dots, t_{im} \quad (1)$$

Each document in the retrieved results ( $D_i$ ) consists of vector of selected terms or phrases ( $t_{im}$ ). The terms and phrases that exceed the threshold are included in the vector as the input for semantic association rules.

## 4 Evaluation

In this section, we present the data collections used for the experiments, the experimental methods, and the other QE techniques for comparison. To evaluate

SemanQE, we compare it with two other query expansion techniques: 1) Cosine similarity-based, a traditional IR technique for the vector space model, 2) SLIPPER, a rule-based query expansion. Performance of these techniques is measured by F-measure and P@20. The data used for experiments are retrieved from TREC via the Lemur search engine.

#### 4.1 Data Collection

The Text Retrieval Conference (TREC) is sponsored by both the National Institute of Standards and Technology (NIST) and U.S. Department of Defense. NIST TREC Document Databases, hereafter called TREC data, are distributed for the development and testing of IR systems and related natural language processing research. The document collections consist of the full text of various newspaper and newswire articles plus government proceedings. The documents have been used to develop a series of large IR test collections known as the TREC collections.

The format of the documents on the TREC disks is a labeled bracketing expressed in the style of SGML. The different datasets on the disks have identical major structures but have different minor structures. Every document is bracketed by <DOC></DOC> tags and has a unique document identifier, bracketed by <DOCNO></DOCNO> tags.

Table 2 show the statistics of records contained in three disks respectively.

**Table 2.** Statistics of TREC Disk 5

Data Description	Size of Dataset
Foreign broadcast information service	Approx. 130,000 documents Approx. 470 MB
Los Angeles Times (from 1989 to 1990)	Approx. 130,000 documents Approx. 475 MB

#### 4.2 Cosine Similarity Model

There are a number of different ways to compute the similarity between documents such as cosine and correlation-based similarity. In our comparison, we use the cosine similarity-based model which is a proven IR technique in the vector space model. In the case of cosine similarity, two documents are thought of as two vectors in the  $m$  dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the  $m \times n$  ratings matrix in Fig. 2, similarity between items  $i$  and  $j$ , denoted by  $sim(i, j)$  is given by

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2} \quad (2)$$

where “.” denotes the dot-product of the two vectors.

#### 4.3 SLIPPER

We chose SLIPPER to compare the performance of SemanQE in generating queries. SLIPPER is an efficient rule-learning system, which is based on confidence-ruled

boosting, a variant of AdaBoost [3]. SLIPPER learns concise rules such as “*protein AND interacts*” --> *Useful*, which shows that if a document contains both term protein and term interacts, it is declared to be useful. These classification rules generated by SLIPPER are then translated into conjunctive queries in the search engine syntax. For instance, the above rule is translated into a query “protein AND interacts.”

#### 4.4 Experimental Results

We conducted a set of experiments to measure the performance of the four techniques: 1) Cosine similarity, 2) SLIPPER, 3) SemanQE-Base, and 4) SemanQE-Ontologies. Since we are interested in whether ontologies have positive impact on the retrieval performance, we evaluate SemanQE in two different ways: 1) SemanQE with ontologies and 2) SemanQE without ontologies. Fig. 3 shows the results of the performance among these four techniques. The y axis is F-measure. F-measure combines precision and recall in order to provide a single number measurement for information extraction systems (3).

$$F_b = \frac{(b^2+1)PR}{b^2P+R} \tag{3}$$

where P is precision, R is recall, b=0 means  $F = \text{precision}$ ,  $b = \infty$  means  $F = \text{recall}$ , b=1 means recall and precision are equally weighted, b=0.5 means recall is half as important as precision. b=2.0 means recall is twice as important as precision. Because  $0 \leq P, R \leq 1$ , a larger value in the denominator means a smaller value overall.

As shown in Fig. 3, SemanQE-Ontologies outperforms the other three techniques from 9.90% to 10.98% better in F-measure in all five cases. The second best technique is SemanQE-base. The performance of the cosine similarity technique is almost equivalent to the SemanQE-base one. SLIPPER turns out to be ranked fourth.

Table 3 shows the overall performance of the four algorithms executing the query set 1-5 on TREC 5 data. The results indicate the improvements in precision at top twenty ranks (P@20) of each algorithm compared to its preceding algorithm. Among the algorithms, SemanQE with Ontologies in P@20 shows the best improvement among the algorithms.

**Table 3.** Results for TREC 5 with Four Query Expansion Algorithms by P@20

Algorithm	TREC 5
	P@20
Cosine	0.2531
SLIPPER	0.3252
SemanQE+base	0.3368
SemanQE+Ontologies	0.3398

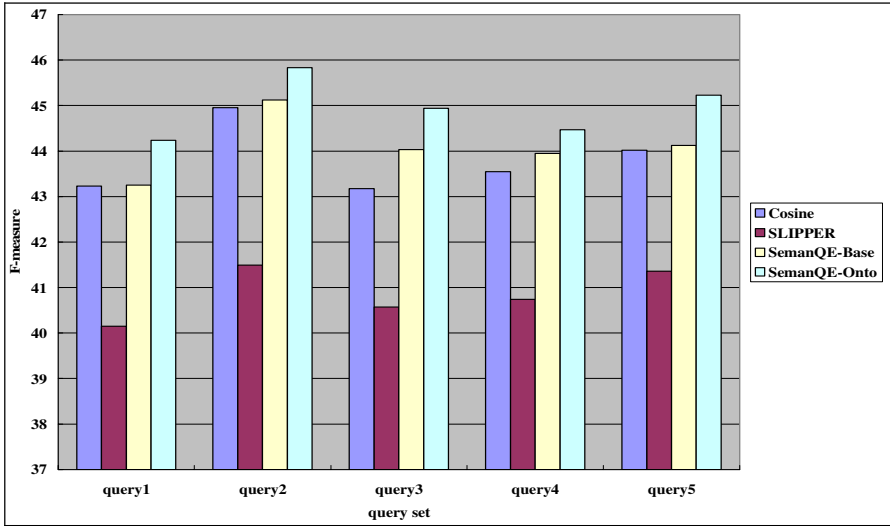


Fig. 3. Performance Comparisons among the Four Techniques

Overall, the results of the experiments show that SemanQE combined with ontologies achieve the best performance in both F-measure and P@20.

## 5 Conclusion

We proposed a novel effective query technique for information extraction, called SemanQE. SemanQE is a hybrid QE technique that applies semantic association rules to the information retrieval problem. Our approach automatically discovers the characteristics of documents that are useful for extraction of a target entity. Using these seed instances, our system retrieves a sample of documents from the database. Then we apply machine learning and information retrieval techniques to learn queries that will tend to match additional useful documents.

Our technique is different from other query expansion techniques in the following aspects. First, it proposes a hybrid query expansion algorithm combining association rules with ontologies and information retrieval techniques. Second, our technique utilizes semantics as well as linguistic properties of unstructured text corpus. Third, our technique makes use of contextual properties of important terms discovered by association rules.

We conducted a series of experiments to examine whether our technique improves the retrieval performance with TREC5 data. We compared our technique, SemanQE+Ontologies with cosine similarity, SLIPPER, and SemanQE without Ontologies. The results show that SemanQE+Ontologies outperforms the other three techniques from 9.2 % to 11.90% better in terms of F-measure in all five cases. In addition, in terms of P@20, SemanQE+Ontologies is significantly better than other technique from 15.49% to 20.98%.

As future studies, we will apply SemanQE to extract entity relations such as protein-protein interactions. We are interested in how SemanQE performs in discovering novel connections among the disjoint literatures where indirect connections exist among the segmented literatures.

## References

1. Agrawal R., Shafer J.C. Parallel mining of association rules, *IEEE Transactions on Knowledge and Data Engineering*, 8 (6): 962-969 1996.
2. Cohen, W.W. and Singer, Y. (1999). Simple, Fast, and Effective Rule Learner, In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, July 18-22, 335-342.
3. French, J.C., Powell, A.L., Gey, F. and Perelman, N. (2001). Exploiting a Controlled Vocabulary to Improve Collection Selection and Retrieval Effectiveness. 10th International Conference on Information and Knowledge Management.
4. Gonzalo, J., Verdejo, F., Chugur, I., Cigarran, J., Indexing with WordNet synsets can improve text retrieval. *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing systems*, Montreal 1998.
5. Lam-Adesina A.M., and Jones, G.J.F. Applying Summarization Techniques for Term Selection in Relevance Feedback, *Proceedings of the 24th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*: 1-9, 2001.
6. Liu, S., Liu, F., Yu, C., and Meng, W. An Effective Approach to Document Retrieval via Utilizing WordNet and Recognizing Phrases, *Proceedings of the 27th annual international Conference on Research and development in Information Retrieval*: 266-272, 2004.
7. Latiri, C.C., Yahia, S.B., Chevallet, J.P., Jaoua, A. *Query expansion using fuzzy association rules between terms*, in JIM'2003, France, September 3-6, 2003.
8. Mihalcea, R., and Moldovan, D. Semantic Indexing Using WordNet Senses. *ACL Workshop on IR & NLP*, 2000.
9. Mitra, C.U., Singhal, A., and Buckley, C. Improving Automatic Query Expansion, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*: 206-214, 1998.
10. Salton, G., Buckley, C., and Fox, E.A. (1983). Automatic query formulations in information retrieval. *Journal of the American Society for Information Science*, 34(4):262-280, July 1983.
11. Sanderson, M. Word sense disambiguation and information retrieval. In *Proceedings, ACM Special Interest Group on Information retrieval*, pages 142-151, 1994.
12. Voorhees, E.M. (1998). Using WordNet for Text Retrieval. In *WordNet, an Electronic Lexical Database*, C. Fellbaum (ed.), MIT Press, 285-303.

# Maintenance of Generalized Association Rules Under Transaction Update and Taxonomy Evolution

Ming-Cheng Tseng<sup>1</sup>, Wen-Yang Lin<sup>2</sup>, and Rong Jeng<sup>1</sup>

<sup>1</sup> Institute of Information Engineering, I-Shou University, Kaohsiung 840, Taiwan  
clark.tseng@msa.hinet.net, rjeng@isu.edu.tw

<sup>2</sup> Dept. of Comp. Sci. & Info. Eng., National University of Kaohsiung, Kaohsiung 811, Taiwan  
wylin@nuk.edu.tw

**Abstract.** Mining generalized association rules among items in the presence of taxonomies has been recognized as an important model in data mining. Earlier work on mining generalized association rules ignore the fact that the taxonomies of items cannot be kept static while new transactions are continuously added into the original database. How to effectively update the discovered generalized association rules to reflect the database change with taxonomy evolution and transaction update is a crucial task. In this paper, we examine this problem and propose a novel algorithm, called IDTE, which can incrementally update the discovered generalized association rules when the taxonomy of items is evolved with new transactions insertion to the database. Empirical evaluations show that our algorithm can maintain its performance even in large amounts of incremental transactions and high degree of taxonomy evolution, and is more than an order of magnitude faster than applying the best generalized associations mining algorithms to the whole updated database.

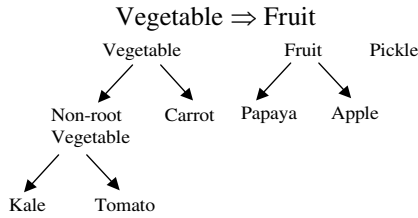
## 1 Introduction

Mining association rules from a large database of business data, such as transaction records, has been a popular topic within the area of data mining [1, 2]. An association rule is an expression of the form  $X \Rightarrow Y$ , where  $X$  and  $Y$  are sets of items. Such a rule reveals that transactions in the database containing items in  $X$  tend to contain items in  $Y$ , and the probability, measured as the fraction of transactions containing  $X$  also containing  $Y$ , is called the *confidence* of the rule. The *support* of the rule is the fraction of the transactions that contain all items in both  $X$  and  $Y$ . For an association rule to be valid, the rule should satisfy a user-specified minimum support, called *ms*, and minimum confidence, called *mc*, respectively.

In many applications, there are taxonomies (hierarchies), explicitly or implicitly, over the items. It may be more useful to find associations at different levels of the taxonomies than only at the primitive concept level [8, 14]. For example, consider the taxonomies of items in Fig. 1. It is likely to happen that the association rule,

Carrot  $\Rightarrow$  Apple (Support = 30%, Confidence = 60%),

does not hold when the minimum support is set to 40%, but the following association rule may be valid.



**Fig. 1.** An example of taxonomies

Up to date, all work on mining generalized association rules, to our best knowledge, confined the taxonomies of items to be static, ignoring the fact that the taxonomy may change as time passes while new transactions are continuously added into the original database [7]. For example, items corresponding to new products have to be added into the taxonomy, and whose insertion would further introduce new classifications if they are of new invented types. On the other hand, items and/or their classifications will also be abandoned if they do not be produced any more. All of these changes would reshape the taxonomy, and in turn would invalidate previously discovered and/or introduce new generalized associations rules, no mention the change caused by the transaction update to the database. Under these circumstances, how to update the discovered generalized association rules effectively becomes a critical task. In this paper, we examine this problem and propose an algorithm called IDTE (Incremental Database with Taxonomy Evolution) which is capable of effectively reducing the number of candidate sets and database re-scanning, and so can update the generalized association rules efficiently. Empirical evaluations show that our algorithm can maintain its performance even at relative low support thresholds, large amounts of incremental transactions, and high degree of taxonomy evolution, and is more than an order of magnitude faster than applying the best generalized associations mining algorithms to the whole updated database.

The remaining of this paper is organized as follows. We discuss related work in Section 2, and describe the problem in Section 3. Detail description of the IDTE algorithm is given in Section 4. In Section 5, we evaluate the performance of the proposed IDTE algorithm. Finally, we conclude the work of this paper in Section 6.

## 2 Related Work

The problem of mining association rules in the presence of taxonomy information was first introduced in [8] and [14], independently. In [14], the problem aimed at finding associations among items at any level of the taxonomy, while in [8], the objective was to discover associations of items in a progressively level-by-level fashion along the taxonomy. The problem of updating association rules incrementally was first addressed by Cheung et al. [4], whose work was later be extended to incorporate the situations of deletion and modification [6]. Since then, a number of techniques have been proposed to improve the efficiency of incremental mining algorithm [9, 10, 13, 15]. But all of them were confined to mining associations among primitive items. Cheung et al. [5] were the first to consider the problem of maintaining generalized (multi-level) asso-

ciation rules. We then extended the problem model to that adopting non-uniform minimum support [16]. To our knowledge, no work to date has considered the issue of maintaining generalized associations while the taxonomy is evolving with the transaction update.

### 3 Problem Statement

In real business applications the database are changing over time; new transactions (may consist of new types of items) are continuously added, while outdated transactions (may consist of abandoned items) are deleted, and the taxonomy that represents the classification of items are also evolved to reflect such changes. This implies that if the updated database is processed afresh, the previously discovered associations might be invalid and some undiscovered associations should be generated. That is, the discovered association rules must be updated to reflect the new circumstance. Analogous to mining associations, this problem can be reduced to updating the frequent itemsets.

#### 3.1 Problem Description

Consider the task of mining generalized frequent itemsets from a given transaction database  $DB$  with the item taxonomy  $T$ . In the literature, although different proposed methods have different strategies in the implementation aspect, the main process involves adding to each transaction the generalized items in the taxonomy. For this reason, we can view the task as mining frequent itemsets from the extended database  $ED$ , the extended version of  $DB$  by adding to each transaction the ancestors of each primitive item in  $T$ . We use  $L^{ED}$  denote the set of discovered frequent itemsets.

Now let us consider the situation when new transactions in  $db$  are added to  $DB$  and the taxonomy  $T$  is changed into a new one  $T'$ . Following the previous paradigm, we can view the problem as follows. Let  $ED'$  and  $ed'$  denote the extended version of the original database  $DB$  and incremental database  $db$ , respectively, by adding to each transaction the generalized items in  $T'$ . Further, let  $UE'$  be the updated extended database containing  $ED'$  and  $ed'$ , i.e.,  $UE' = ED' + ed'$ . The problem of updating  $L^{ED}$  when new transactions  $db$  are added to  $DB$  and  $T$  is changed into  $T'$  is equivalent to finding the set of frequent itemsets in  $UE'$ , denoted as  $L^{UE'}$ .

#### 3.2 Situations for Taxonomy Evolution and Frequent Itemsets Update

In this subsection, we will describe different situations for taxonomy evolution, and clarify the essence of frequent itemsets update for each type of taxonomy evolutions. According to our observation, there are four basic types of item updates that will cause taxonomy evolution: item insertion, item deletion, item rename and item reclassification. Each of them will be elaborated in the following. For simplicity, in all figures hereafter item “A” stands for “Vegetable”, “B” for “Non-root Vegetable”, “C” for “Kale”, “D” for “Carrot”, “E” for “Tomato”, “F” for “Fruit”, “G” for “Papaya”, “H” for “Apple”, “I” for “Pickle”, “J” for “Root Vegetable”, “K” for “Potato”, “B1” for “Non-root Vegetable New”, and “G1” for “Papaya New”.



**Type 1: Item Insertion.** The strategies to handle this type of update operation are different, depending on whether an inserted item is primitive or generalized. When the new inserted item is primitive, we do not have to process it until an incremental database update containing that item indeed occurs. This is because the new item does not appear in the original database, neither in the discovered associations. However, if the new item is a generalization, then the insertion will affect the discovered associations since a new generalization often incurs some item reclassification. Fig. 2 shows this type of taxonomy evolution, where a new item “K” is inserted as a primitive item and “B” is a generalized item.

**Type 2: Item Deletion.** Unlike the case of item insertion, the deletion of a primitive item from the taxonomy would incur inconsistency problem. In other words, if there is no transaction update to delete the occurrence of that item, then the refined item taxonomy will not conform to the updated database. An outdated item still appears in the transaction of interest! To simplify the discussion, we assume that the evolution of the taxonomy is always consistent with the transaction update to the database. Additionally, the removal of a generalization may also lead to item reclassification. So we always have to deal with the situation caused by item deletion. Fig. 3 shows this type of taxonomy evolution, where a primitive item “C” and a generalized item “B” are deleted respectively.

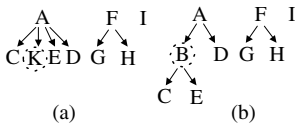


Fig. 2. Item insertion

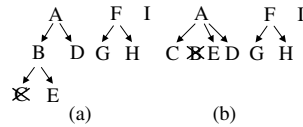


Fig. 3. Item deletion

**Type 3: Item Rename.** When items are renamed, we do not have to process the database; we just replace the frequent itemsets with new names since the process codes of renamed items are the same. Fig. 4 shows this type of taxonomy evolution, where items “G” and “B” are renamed to “G1” and “B1”, respectively.

**Type 4: Item Reclassification.** Among the four types of taxonomy updates this is the most profound operation. Once an item, primitive or generalized, is reclassified into another category, all of its ancestor (generalized items) in the old and the new taxonomies are affected. For example, in Fig. 5, the two shifted items “E” and “G” will affect the support counts of itemsets containing “A”, “B”, or “F”.

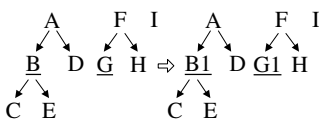


Fig. 4. Item rename

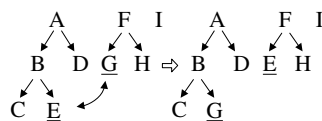


Fig. 5. Item reclassification

## 4 The Proposed Method

A straightforward method to update the discovered generalized frequent itemsets would be to run any of the algorithms for finding generalized frequent itemsets, such as Cumulate and Stratify [14], on the updated extended database  $UE'$ . This simple way, however, does not utilize the discovered frequent itemsets and ignores the fact that scanning the whole updated database would be avoided. Instead, a better approach is to, within the set of discovered frequent itemsets  $L^{ED}$ , differentiate the itemsets that are unaffected with respect to the taxonomy evolution from the others, and then utilize them to avoid unnecessary computation in the course of incremental update. To this end, we first have to identify the unaffected items whose supports does not change with respect to the taxonomy evolution, and then use them to identify the unaffected itemsets. We introduce the following notation to facilitate the discussion:  $I, J$  denote the set of primitive items and the set of generalized items in  $T$ , respectively, and  $I', J'$  represent the counterparts in  $T'$ .

**Definition 1.** An item in  $T$  is called an *unaffected item* if its support does not change with respect to a taxonomy evolution.

**Lemma 1.** Consider a primitive item  $a$  in  $T \cup T'$ . Then

- (a)  $sup_{ED'}(a) = sup_{ED}(a)$  if  $a \in I \cap I'$ , and
- (b)  $sup_{ED'}(a) = 0$  if  $a \in I' - I$ ,

where  $sup_{ED'}(a)$  and  $sup_{ED}(a)$  denote the supports of item  $a$  in  $ED'$  and  $ED$ , respectively.

**Lemma 2.** Consider a generalized item  $g$  in  $T'$ . Then  $sup_{ED'}(g) = sup_{ED}(g)$  if  $des_{T'}(g) = des_T(g)$ , where  $des_{T'}(g)$  and  $des_T(g)$  denote the sets of descendant primitive items of  $g$  in  $T'$  and  $T$ , respectively.

In summary, Lemmas 1 and 2 state that an item is unaffected by the taxonomy evolution if it is a primitive item before and after the taxonomy evolution or it is a generalized item whose descendant set of primitive items remains the same.

**Definition 2.** An itemset  $A$  in  $ED'$  is called an *unaffected itemset* if its support does not change with respect to a taxonomy evolution.

**Lemma 3.** Consider an itemset  $A$  in  $ED'$ . Then

- (a)  $sup_{ED'}(A) = sup_{ED}(A)$  if  $A$  contains unaffected items only; or
- (b)  $sup_{UE'}(A)$  does not exist if  $A$  contains at least one item  $a$ , for  $a \in I - I'$ .

Now that we have clarified how to identify the unaffected itemsets, we will further show how to utilize this information to alleviate the overhead in updating the supports of itemsets. Consider a candidate itemset  $A$  generated during the mining process. We observe that there are six different cases in arriving at the support counts of  $A$  in the whole updated database  $UE'$ .

- (1) If  $A$  is an unaffected itemset and is frequent in  $ed'$  and  $ED'$ , then it is also frequent in the updated extended database  $UE'$ .
- (2) If  $A$  is an unaffected itemset and is infrequent in  $ed'$  and  $ED'$ , then it is also infrequent in  $UE'$ .

- (3) If  $A$  is an unaffected itemset and is infrequent in  $ed'$  but frequent in  $ED'$ , then a simple calculation can determine whether  $A$  is frequent or not in  $UE'$ .
- (4) If  $A$  is an unaffected itemset and is frequent in  $ed'$  but infrequent in  $ED'$ , then it is an undetermined itemset in  $UE'$ , i.e., it may be frequent or infrequent.
- (5) If  $A$  is not unaffected and is frequent in  $ed'$ , then it is an undetermined itemset in  $UE'$ .
- (6) If  $A$  is not unaffected and is infrequent in  $ed'$ , then it is an undetermined itemset in  $UE'$ .

Note that only Cases 4 to 6 requires an additional scan of  $ED'$  to determine the support count of  $A$  in  $UE'$ . For Case 4, after scanning  $ed'$  and comparing with  $ms$ , if  $A$  is frequent in  $ed'$ ,  $A$  may become frequent in  $UE'$ . Then we need rescan  $ED'$  to determine the support count of  $A$ . For Cases 5 and 6, since  $A$  is not an unaffected itemset its support count would be changed in  $ED'$ . Therefore, we need further scan  $ED'$  to decide whether it is frequent or not.

For Cases 1 to 3, there is no need to further scan  $ED'$  to determine the support counts of itemset  $A$ . That is, we have utilized the information of unaffected itemsets and discovered frequent itemsets to avoid such a database scan. Furthermore, the identification of itemsets satisfies Case 2 provides another opportunity for candidate pruning.

The IDTE algorithm is shown in Fig. 6. An example for illustrating the proposed IDTE algorithm is provided in Fig 7, where  $ms = 20\%$ .

---

1.  $k = 1$ ;
2. **repeat**
3.     **if**  $k = 1$  **then** generate  $C_1$  from  $T'$ ;
4.     **else**  $C_k = \text{apriori-gen}(L_k^{UE'})$ ;
5.     Delete any candidate in  $C_k$  that consists of an item and its ancestor;
6.     Load original frequent  $k$ -itemsets  $L_k^{ED}$ ;
7.     Divide  $C_k$  into two subsets:  $C_X$  and  $C_Y$ ; /\*  $C_X$  consists of unaffected itemsets in  $ED'$ , and  $C_Y = C_k - C_X$ . \*/
8.     Divide  $C_X$  into two subsets:  $C_{Xa}$  and  $C_{Xb}$ ; /\*  $C_{Xa}$  consists of frequent itemsets in  $L_k^{ED}$ , and  $C_{Xb} = C_X - C_{Xa}$ . \*/
9.     Scan  $ed'$  to count  $\text{sup}_{ed'}(A)$  for each itemset  $A$  in  $C_k$ ;
10.      $L_k^{ed'} = \{A \mid A \in C_k \text{ and } \text{sup}_{ed'}(A) \geq ms\}$ ;
11.     Delete any candidate  $A$  from  $C_{Xb}$  if  $A \notin L_k^{ed'}$ ; /\* Case 2 \*/
12.     Scan  $ED'$  to count  $\text{sup}_{ED'}(A)$  for each itemset  $A$  in  $C_{Xb}$  and  $C_Y$  having no new primitive item; /\* Case 4, 5 & 6, and Lemma 1(b) \*/
13.     Calculate  $\text{sup}_{UE'}(A)$  for each itemset  $A$  in  $C_k$ ;
14.      $L_k^{UE'} = \{A \mid A \in C_k \text{ and } \text{sup}_{UE'}(A) \geq ms\}$ ;
15. **until**  $L_k^{UE'} = \emptyset$
16.     Result =  $\bigcup_k L_k^{UE'}$ ;

---

**Fig. 6.** Algorithm IDTE

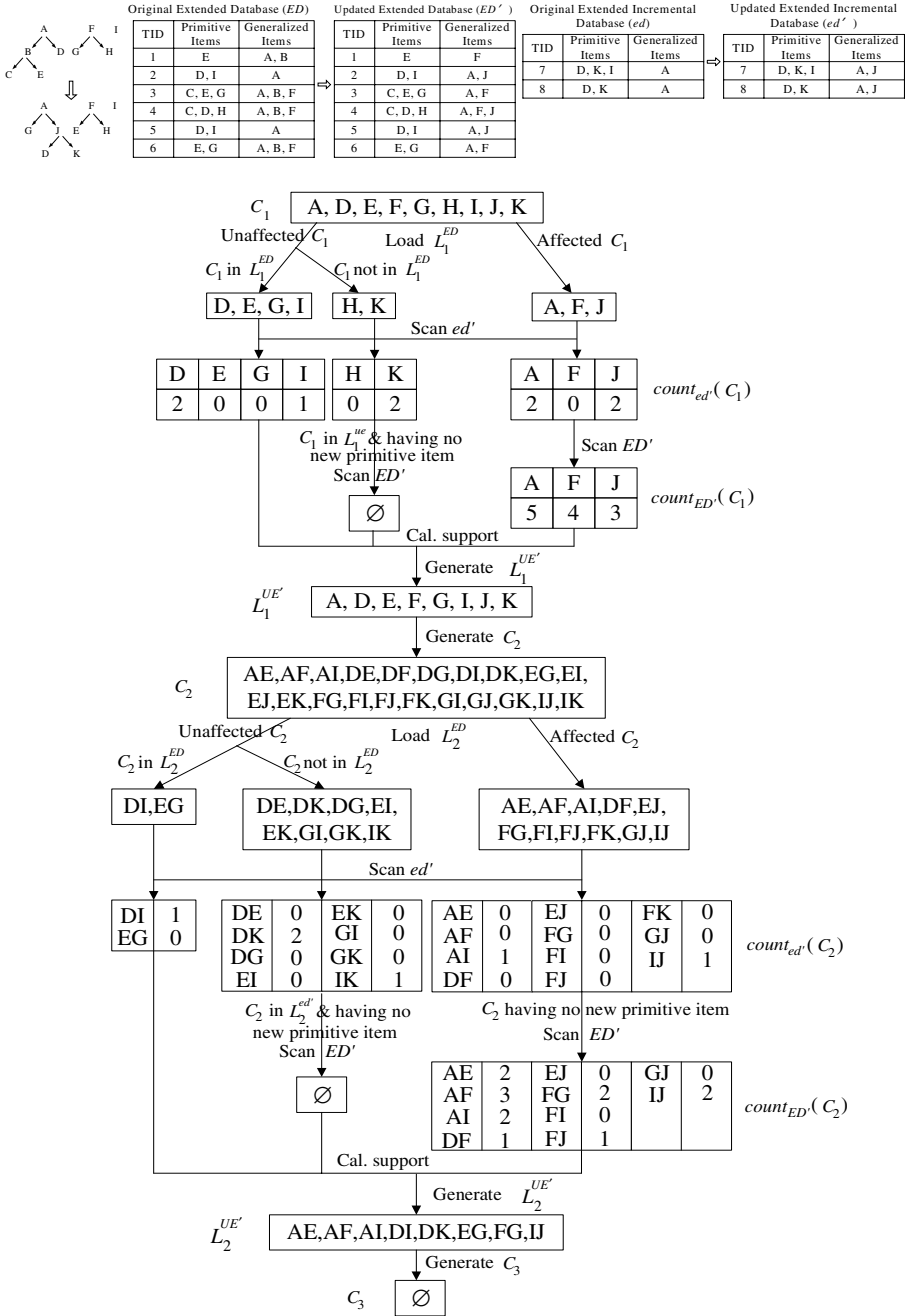


Fig. 7. Illustration of algorithm IDTE

## 5 Experiments

In order to examine the performance of IDTE, we conducted experiments to compare its performance with that of applying generalized association mining algorithms, including Cumulate and Stratify, to the whole updated database. Synthetic datasets generated by the IBM data generator [2] were used in the experiments. The parameter settings for synthetic data are shown in Table 1. The comparisons were evaluated from different aspects: include minimum support, incremental transaction size, fanout, number of groups, and percent of affected items, i.e., the ratio of affected items to total items. In the implementation of each algorithm, we also adopted two different support counting strategies: one with the horizontal counting [1, 2, 3, 11] and the other with the vertical intersection counting [12, 17]. For the horizontal counting, the algorithms are denoted as Cumulate(H), Stratify(H), and IDTE(H) while for the vertical intersection counting, the algorithms are denoted as Cumulate(V), Stratify(V), and IDTE(V). All experiments were performed on an Intel Pentium-IV 2.80GHz with 2GB RAM, running on Windows 2000.

**Minimum Supports:** We first compared the performance of these three algorithms with varying minimum supports at 40,000 incremental transactions with constant affected item percent. The experimental results are shown in Fig. 8. As shown in the figure, IDTE perform significantly better than Cumulate and Stratify. Besides, algorithms with vertical counting strategy are better than their counterpart with horizontal strategy.

**Transaction Sizes:** We then compared the three algorithms under varying transaction sizes at  $ms = 1.0\%$  with constant affected item percent. As the results shown in Fig. 9, the running time of all algorithms increase in proportional to the incremental size. Furthermore, IDTE(H) significantly outperforms Cumulate(H) and Stratify(H) and similarly, IDTE(V) beats Cumulate(V) and Stratify(V).

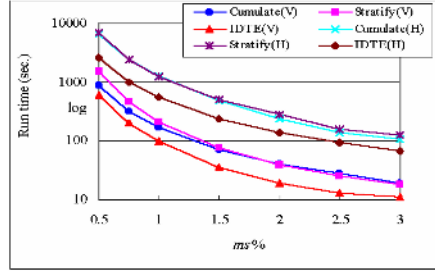
**Fanout:** We changed the fanout from 3 to 11 at  $ms = 1.0\%$  with constant affected item percent and 40,000 incremental transactions. The experimental results are shown in Fig. 10. It can be observed that all algorithms perform faster as the fanout increases because the number of generalized items decreases upon increasing the number of fanout. Again, IDTE significantly outperforms Cumulate and Stratify, either with vertical or horizontal counting strategy.

**Number of Groups:** We varied the number of groups from 15 to 35 at  $ms = 1.0\%$  with constant affected item percent and 40,000 incremental transactions. As Fig. 11 shows, the effect of increasing the number of groups is similar to that of increasing the fanout. The reason is that the number of items within a specific group decreases as the number of groups increases, so the probability of a generalized item decreases.

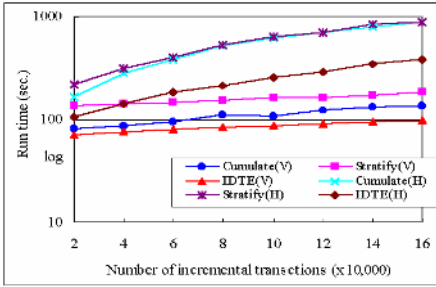
**Affected Item Percent:** We finally compared the three algorithms under varying affected item percent at  $ms = 1.0\%$  and 40,000 incremental transactions. The affected items were randomly chosen, undergoing reclassification. The results are depicted in Fig. 12.

**Table 1.** Parameter settings

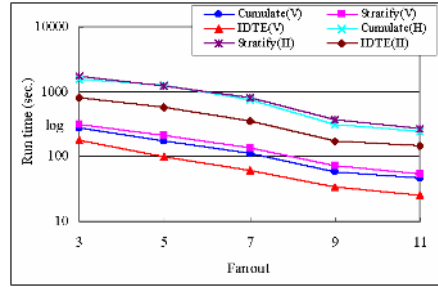
Parameter	Default value	
$ D $	Number of original transactions	177,783
$ db $	Number of incremental transactions	160,000
$ t $	Average size of transactions	16
$N$	Number of items	231
$R$	Number of groups	30
$L$	Number of levels	3
$F$	Fanout	5



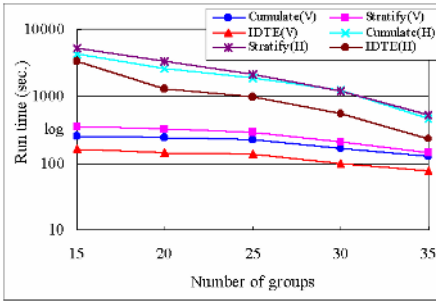
**Fig. 8.** Different  $ms$



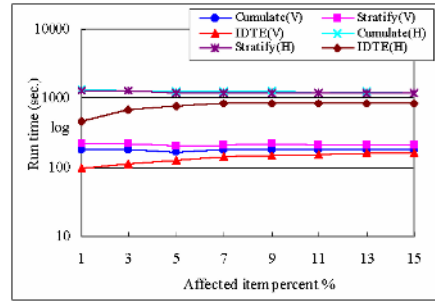
**Fig. 9.** Different transactions



**Fig. 10.** Varying fanout



**Fig. 11.** Varying number of groups



**Fig. 12.** Different affected item percent

In summary, we observe that IDTE(V) performs better than Cumulate(V) and Stratify(V) while IDTE(H) performs better than Cumulate(H) and Stratify(H) in all aspects of evaluation. Besides, all algorithms with vertical support counting strategy perform better than their counterpart with horizontal counting strategy.

## 6 Conclusions

We have investigated in this paper the problem of updating generalized association rules when new transactions are inserted into the database and the taxonomy of items is evolved over time. We also have presented a novel algorithm, IDTE, for updating gen-

eralized frequent itemsets. Empirical evaluation on synthetic data showed that the IDTE algorithm is very efficient, which outperforms applying the best generalized association mining algorithms to the whole updated database. In the future, we will extend the problem of updating generalized association rule to a more general model that adopts non-uniform minimum support to solve the problem that new introduced items usually have much lower supports.

## References

1. Agrawal R., Imielinski T., Swami A.: Mining Association Rules between Sets of Items in Large Databases. In Proc. 1993 ACM-SIGMOD Intl. Conf. Management of Data (1993) 207-216
2. Agrawal R., Srikant R.: Fast Algorithms for Mining Association Rules. In Proc. 20th Intl. Conf. Very Large Data Bases (1994) 487-499
3. Brin S., Motwani R., Ullman J.D., Tsur S.: Dynamic Itemset Counting and Implication Rules for Market Basket Data. SIGMOD Record, Vol. 26 (1997) 255-264
4. Cheung D.W., Han J., Ng V.T., Wong C.Y.: Maintenance of Discovered Association Rules in Large Databases: An Incremental Update Technique. In Proc. 1996 Int. Conf. Data Engineering (1996) 106-114
5. Cheung D.W., Ng V.T., Tam B.W.: Maintenance of Discovered Knowledge: A case in Multi-level Association Rules. In Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (1996) 307-310
6. Cheung D.W., Lee S.D., Kao B.: A General Incremental Technique for Maintaining Discovered Association Rules. In Proc. DASFAA'97 (1997) 185-194
7. J. Han, Y. Fu: Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases. In Proc. AAAI'94 Workshop on Knowledge Discovery in Databases (KDD'94) (1994) 157-168
8. Han J., Fu Y.: Discovery of Multiple-level Association Rules from Large Databases. In Proc. 21st Intl. Conf. Very Large Data Bases, Zurich, Switzerland (1995) 420-431
9. Hong T.P., Wang C.Y., Tao Y.H.: Incremental Data Mining Based on Two Support Thresholds. In Proc. 4 Int. Conf. Knowledge-Based Intelligent Engineering Systems and Allied Technologies (2000) 436-439
10. Ng K.K., Lam W.: Updating of Association Rules Dynamically. In Proc. 1999 Int. Symp. Database Applications in Non-Traditional Environments (2000) 84-91
11. Park J.S., Chen M.S., Yu P.S.: An Effective Hash-based Algorithm for Mining Association Rules. In Proc. 1995 ACM SIGMOD Intl. Conf. on Management of Data, San Jose, CA, USA (1995) 175-186
12. Savasere A., Omiecinski E., Navathe S.: An Efficient Algorithm for Mining Association Rules in Large Databases. In Proc. 21st Intl. Conf. Very Large Data Bases (1995) 432-444
13. Sarda N.L., Srinivas N.V.: An Adaptive Algorithm for Incremental Mining of Association Rules. In Proc. 9th Int. Workshop on Database and Expert Systems Applications (DEXA'98) (1998) 240-245
14. Srikant R., Agrawal R.: Mining Generalized Association Rules. In Proc. 21st Int. Conf. Very Large Data Bases (1995) 407-419
15. Thomas S., Bodagala S., Alsabti K., Ranka S.: An Efficient Algorithm for the Incremental Update of Association Rules in Large Databases. In Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (1997)
16. Tseng M.C., Lin W.Y.: Maintenance of Generalized Association Rules with Multiple Minimum Supports. Intelligent Data Analysis, Vol. 8 (2004) 417-436
17. Zaki M.J.: Scalable Algorithms for Association Mining. IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 2 (2000) 372-390

# Prince: An Algorithm for Generating Rule Bases Without Closure Computations\*

T. Hamrouni, S. Ben Yahia, and Y. Slimani

Département des Sciences de l'Informatique,  
Campus Universitaire, 1060 Tunis, Tunisie  
{s dok.benyahia, yahya.slimani}@fst.rnu.tn

**Abstract.** The problem of the relevance and the usefulness of extracted association rules is becoming of primary importance, since an overwhelming number of association rules may be derived, even from reasonably sized databases. To overcome such drawback, the extraction of reduced size generic bases of association rules seems to be promising. Using the concept of minimal generator, we propose an algorithm, called PRINCE, allowing a shrewd extraction of generic bases of rules. To this end, PRINCE builds the partial order. Its originality is that this partial order is maintained between minimal generators and no more between closed itemsets. A structure called *minimal generator lattice* is then built, from which the derivation of the generic association rules becomes straightforward. An intensive experimental evaluation, carried out on benchmarking sparse and dense datasets, showed that PRINCE largely outperforms the pioneer level-wise algorithms, *i.e.*, CLOSE, A-CLOSE and TITANIC.

**Keywords:** Data mining, Formal Concept Analysis, generic rule bases, minimal generator lattice.

## 1 Introduction

The last decade was marked by an "obsessional" algorithmic effort to reduce the computation time of the interesting pattern extraction step. The obtained success is primarily due to prowesses of programming conjuncted with intensive handling of compact data structures in the main memory. However, it seems obvious that this frenzy made loses sight of the essential objective of this step, *i.e.*, to extract a reliable knowledge, of exploitable size for end-users. Thus, almost all these algorithms focused on enumerating maximal or closed patterns – presenting a frequency of appearance considered to be satisfactory [1]. As a drawback of this success, this enumeration will generate an impressive and not exploitable number of association rules, even for a reasonable context size. In this situation, the approach based on the extraction of closed patterns (or itemsets) [2] presented a clear promise to reduce considerably the association rule list size. This approach, relying on the Formal Concept Analysis (FCA) mathematical background [3],

---

\* A French version was previously published in French at INFORSID'2005: <http://inforsid2005.imag.fr/index.htm>.



proposes to reduce the search space by detecting intrinsic structural properties. Thus, the use of monotonous and non monotonous constraints, during the browsing of the search space, is a typical behavior of this approach [4]. Therefore, the problem of mining association rules might be reformulated, under the (frequent) closed itemsets discovery point of view, as the following two steps [4]:

1. Discover two distinct "closure systems", *i.e.*, sets of sets which are closed under the intersection operator, namely the set of closed itemsets and the set of minimal generators. Also, the *upper cover* ( $Cov^u$ ) of each closed itemset should be available.
2. From all the discovered information during the first step, *i.e.*, both closure systems and the upper cover sets, derive generic bases of association rules (from which all the remaining rules can be derived).

The essential report after an overview of the state of the art of algorithms [2,5,6,7,8] aiming to discover closed itemsets can be summarized in what follows:

1. Modest performances on sparse contexts. Indeed, computing itemset closures in this type of contexts is heavily handicapping these algorithm performances.
2. Negligence of maintaining the order covering the relationship between the closed itemsets.

In this paper, we propose a new algorithm, called PRINCE, aiming to extract generic bases of association rules. PRINCE performs a level-wise browsing of the search space. Its main originality is that it is the only one who accepted to bear the cost of building the partial order. Interestingly enough, to amortize this prohibitive cost, the partial order is maintained between minimal generators and not between closed itemsets. Hence, itemset closures are not computed but derived when PRINCE performs a simple sweeping of the partially ordered structure to derive generic bases of association rules. Experimental results, carried out on typical benchmark datasets, are very encouraging and showed that this astute partial order construction is not handicapping. Indeed, PRINCE performances largely outperformed those of well known level-wise browsing algorithms, *i.e.*, CLOSE [2], A-CLOSE [5], and TITANIC [7]. These last determine at least the "key" information provided by the minimal generator set. Due to space limit, we report our results only on two datasets.

The remainder of the paper is organized as follows. Section 2 presents the basic mathematical foundations for the derivation of generic bases of association rules. We describe related work and motivations in section 3. Section 4 is dedicated to the presentation of the algorithm PRINCE. Experimental results showing the utility of the proposed approach are presented in section 5. The conclusion and future work are presented in section 6.

## 2 Mathematical Background

Due to lack of available space, interested reader for key results from the Galois lattice-based paradigm in FCA is referred to [9].

**Frequent Closed Itemset:** An itemset  $I \subseteq \mathcal{I}$  is said to be *closed* if  $\omega(I) = I$  <sup>(1)</sup> [2].  $I$  is said to be *frequent* if its *relative support*,  $\text{Supp}(I) = \frac{|\psi(I)|}{|\mathcal{O}|}$ , exceeds a user-defined minimum threshold, denoted *minsup*. In the remainder, we will use the absolute support.

**Minimal Generator** [10]: An itemset  $g \subseteq \mathcal{I}$  is said to be *minimal generator* of a closed itemset  $f$ , if and only if  $g'' = f$  and does not exist  $g_1 \subseteq g$  such that  $g_1'' = f$ . The set  $MG_f$  of the minimal generators of  $f$  is:  $MG_f = \{g \subseteq \mathcal{I} \mid g'' = f \wedge \nexists g_1 \subset g \text{ such as } g_1'' = f\}$ .

**Iceberg Galois Lattice:** When only frequent closed itemsets are considered with set inclusion, the resulting structure  $(\hat{\mathcal{L}}, \subseteq)$  only preserves the *Join* operator [9]. This is called a *join semi-lattice* or *upper semi-lattice* and referred to as "*Iceberg Galois Lattice*" [7]

**Minimal Generator Lattice:** A *minimal generator lattice* is equivalent to an Iceberg Galois lattice such as each equivalence class contains only the associated minimal generators [11].

### 3 Related Work and Motivations

Since the apparition of the approach based on the extraction of the frequent closed itemsets [2], several generic bases were introduced among which those of Bastide *et al.* [10] and which are defined as follows:

1. The *generic basis for exact association rules* is defined as follows:

**Definition 1.** Let  $\mathcal{FCI}_{\mathcal{K}}$  be the set of frequent closed itemsets extracted from the extraction context  $\mathcal{K}$ . For each frequent closed itemset  $f \in \mathcal{FCI}_{\mathcal{K}}$ , let  $MG_f$  be the set of its minimal generators. The generic basis of exact association rules  $GB$  is given by:  $GB = \{R: g \Rightarrow (f - g) \mid f \in \mathcal{FCI}_{\mathcal{K}} \text{ and } g \in MG_f \text{ and } g \neq f^{(2)}\}$ .

2. The *transitive reduction of the informative base* [10] is defined as follows:

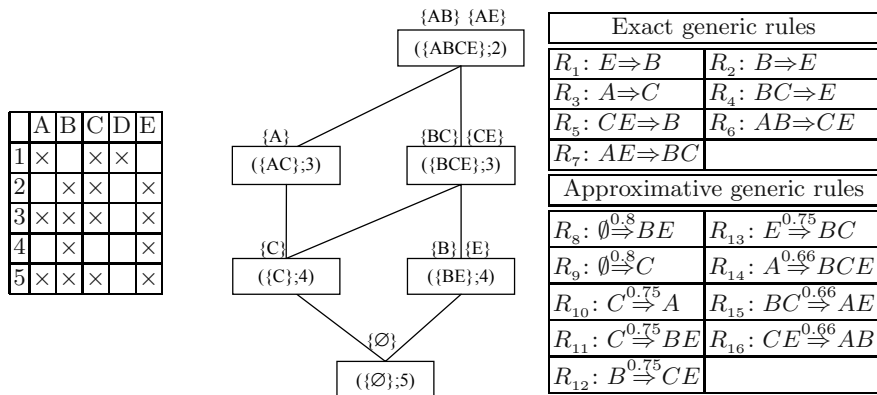
**Definition 2.** Let  $\mathcal{FMG}_{\mathcal{K}}$  be the set of frequent minimal generators extracted from the extraction context  $\mathcal{K}$ . The transitive reduction  $RI$  is given by:  $RI = \{R \mid R: g \Rightarrow (f - g) \mid f \in \mathcal{FCI}_{\mathcal{K}} \text{ and } g \in \mathcal{FMG}_{\mathcal{K}} \text{ and } g'' \prec f^{(3)} \text{ and } \text{Conf}(R) \geq \text{minconf}\}$ .

In the remainder of the paper, we will refer to the generic association rules formed by the couple  $(GB, RI)$ . This couple is informative, sound and lossless [10,12] and the rules forming it are referred as *informative association rules*. Thus, given an Iceberg Galois lattice – in which each frequent closed itemset is decorated by its list of the minimal generators – the derivation of these rules can

<sup>1</sup> In the remainder, the closure operator  $\omega$  is indicated by  $''$ .

<sup>2</sup> The condition  $g \neq f$  ensures discarding non-informative rules of the form  $g \Rightarrow \emptyset$ .

<sup>3</sup> The notation  $\prec$  indicates that  $f$  covers  $g''$  in  $(\hat{\mathcal{L}}, \subseteq)$ .



**Fig. 1.** Left: Extraction context  $\mathcal{K}$  Center: The associated *Iceberg Galois lattice* for  $minsup=2$ . Right: *Informative association rules* for  $minsup=2$  and  $minconf=0.5$ .

be performed straightforwardly. Indeed, generic approximative rules represent "inter-node" implications, assorted with the confidence measure, between two adjacent comparable equivalence classes, *i.e.*, from a frequent closed itemset to another frequent closed itemset immediately covering it.

According to the definition of the informative association rules, we note that they require the extraction of the frequent closed itemsets and their associated minimal generators as well as the upper cover of each frequent closed itemset. Then, constructing the partially ordered structure is a *sine qua non* condition for obtaining the *RI* basis. Thus, information conveyed by minimal generator set and partial order are of paramount importance.

In order to palliate the insufficiencies of existing frequent closed based algorithms, *i.e.*, the cost of the closure computation as well as neglecting the partial order construction, we will introduce a new algorithm called PRINCE. PRINCE highly reduces the cost of closure computation and generates the partially ordered structure, which makes it able to extract generic rule bases without coupling it with another algorithm.

## 4 Prince Algorithm

PRINCE takes as input an extraction context  $\mathcal{K}$ , the minimum threshold of support  $minsup$  and the minimum threshold of confidence  $minconf$ . It outputs the list of the frequent closed itemsets and their associated minimal generators as well as generic bases of association rules. Thus, PRINCE operates in three successive steps: (i) Minimal generator determination (ii) Partial order construction (iii) Extraction of generic rule bases.

### 4.1 Minimal Generator Determination

Following the "Test-and-generate" technique, PRINCE traverses the search space by level to determine the set of frequent minimal generators  $\mathcal{FMG}_{\mathcal{K}}$  sorted by

decreasing support values as well as the negative border of minimal generators  $\mathcal{GBd}^{-(4)}$ [13]. In the second step, the set of frequent minimal generators will serve as a backbone to construct the *minimal generator lattice*. As in the case of TITANIC, the negative border will be used, in the second step, to determine the support of a non-minimal generator using the following proposition:

**Proposition 1.** [7] *Let  $\mathcal{MG}_{\mathcal{K}} = \mathcal{FMG}_{\mathcal{K}} \cup \mathcal{GBd}^{-}$  be the set of minimal generators extracted from the context  $\mathcal{K}$ . If  $X$  is not a minimal generator then  $Supp(X) = \min \{Supp(g) \mid g \in \mathcal{MG}_{\mathcal{K}} \text{ and } g \subset X\}$ .*

PRINCE uses, in this step, the same pruning strategies introduced in TITANIC namely *minsup*, the ideal order of the minimal generators and the estimated support.

## 4.2 Partial Order Construction

In this step, the frequent minimal generator set  $\mathcal{FMG}_{\mathcal{K}}$  will form a *minimal generator lattice*, and this *without any access to the extraction context*. The main idea is how to construct the partial order without computing itemset closures, *i.e.*, how guessing the subsumption relation by only comparing minimal generators? To achieve this goal, the list of immediate successors<sup>(5)</sup> of each equivalence class will be updated in an iterative way. The processing of the frequent minimal generator set is done according to the order imposed in the first step (*i.e.*, by decreasing support values). Each frequent minimal generator  $g$  of size  $k$  ( $k \geq 1$ ) is introduced into the minimal generator lattice by comparing it to the immediate successors of its  $(k-1)$ -subsets<sup>(6)</sup>. This is based on the isotony property of the closure operator [14]. Indeed, let  $g_1$ , a  $(k-1)$ -itemset, be one of the subsets of  $g$ ,  $g_1 \subseteq g \Rightarrow g_1'' \subseteq g''$ . Thus, the equivalence class to which belongs  $g$  is a successor (not necessarily an immediate one) of the equivalence class to which belongs  $g_1$ .

While comparing  $g$  to the list of the immediate successors of  $g_1$ , noted  $L$ , two cases are to be distinguished. If  $L$  is empty then  $g$  is added to  $L$ . Otherwise,  $g$  is compared to the elements already belonging to  $L$  (*cf.* Proposition 2). The imposed order in the first step allows to distinguish only two cases sketched by Proposition 2 by replacing the frequent minimal generators  $X$  and  $Y$  by respectively  $g$  and one of the elements of  $L$ .

**Proposition 2.** [15] *Let  $X, Y \in \mathcal{FMG}_{\mathcal{K}}$ ,  $\mathcal{C}_X$  and  $\mathcal{C}_Y$  their respective equivalence classes:*

- a. *If  $Supp(X) = Supp(Y) = Supp(X \cup Y)$  then  $X$  and  $Y$  belong to the same equivalence class.*
- b. *If  $Supp(X) < Supp(Y)$  and  $Supp(X) = Supp(X \cup Y)$  then  $\mathcal{C}_X$  (resp.  $\mathcal{C}_Y$ ) is a successor (resp. predecessor) of  $\mathcal{C}_Y$  (resp.  $\mathcal{C}_X$ ).*

<sup>4</sup> An itemset belong to  $\mathcal{GBd}^{-}$  if it is an infrequent minimal generator and all its subsets are frequent minimal generators.

<sup>5</sup> By the term "immediate successor", we indicate a frequent minimal generator, unless otherwise specified.

<sup>6</sup> In the first step and for each candidate of size  $k$ , links towards its subsets of size  $(k-1)$  are stored during the check of the ideal order property.

During these comparisons and to avoid redundant closure computations, PRINCE introduces two complementary functions. These functions make it possible to maintain the concept of equivalence class throughout processing. To this end, each equivalence class  $\mathcal{C}$  will be characterized by a *representative* itemset, which is the *first* frequent minimal generator introduced into the *minimal generator lattice*. Both functions are described below:

1. **MANAGE-EQUIV-CLASS**: This function is used if a frequent minimal generator, say  $g$ , is compared to the representative itemset of its equivalence class, say  $\mathcal{R}$ . The **MANAGE-EQUIV-CLASS** function replaces all occurrences of  $g$  by  $\mathcal{R}$  in the immediate successor lists in which  $g$  was added. Then, comparisons to carry out with  $g$  will be made with  $\mathcal{R}$ . Thus, for each equivalence class, only its representative itemset appears in the lists of the immediate successors.
2. **REPRESENTATIVE**: This function makes it possible to find, for each frequent minimal generator  $g$ , the representative  $\mathcal{R}$  of its equivalence class in order to complete the immediate successors of  $\mathcal{C}_g$ . This allows to manage only one list of the immediate successors for all frequent minimal generators belonging to the same equivalence class.

The pseudo-code of the second step is given by the **GEN-ORDER** procedure (Algorithm 1). Each entry, say  $g$ , in  $\mathcal{FMG}_{\mathcal{K}}$  is composed by the following fields: (i) **support**: the support of  $g$  (ii) **direct-subsets**: the list of the  $(k-1)$ -subsets of  $g$  (iii) **immediate-succs**: the list of the immediate successors of  $g$ .

### 4.3 Extraction of Generic Rule Bases

In this step, PRINCE extracts the *informative association rules*. For this purpose, PRINCE finds the frequent closed itemset corresponding to each equivalence class by using Proposition 3.

**Proposition 3.** [15] *Let  $f$  and  $f_1$  be two closed itemsets such that  $f$  covers  $f_1$  in the Galois lattice  $\mathcal{L}_{\mathcal{K}}$ . Let  $MG_f$  be the set of minimal generators of  $f$ . The closed itemset  $f$  can be composed as follows:  $f = \cup\{g|g \in MG_f\} \cup f_1$ .*

The traversal of the *minimal generator lattice* is carried out in an ascending manner from the equivalence class whose minimal generator is the empty set<sup>(7)</sup> (denoted  $\mathcal{C}_\emptyset$ ) to the non subsumed equivalence class(es). If the closure of the empty set is not null, the informative exact rule between the empty set and its closure is then extracted. Having the partial ordered structure built, PRINCE extracts the informative approximative rules between the empty set and the frequent closed itemsets of the upper cover of  $\mathcal{C}_\emptyset$ . These closures are found, by applying Proposition 3, using the minimal generators of each equivalence class and the closure of the empty set. Equivalence classes forming the upper cover of  $\mathcal{C}_\emptyset$  are stored which makes it possible to apply the same process to them. By the

<sup>7</sup> This class is called the Bottom element of the lattice [9]. The corresponding closure is calculated in the first step by collecting items having a support equal to the number of transactions in the extraction context.

```

1 Procedure: GEN-ORDER
  Data: -  $\mathcal{FMG}_{\mathcal{K}}$ .
  Results: - The elements of  $\mathcal{FMG}_{\mathcal{K}}$  partially ordered in the form of a
    minimal generator lattice.
2 Begin
3   Foreach ( $g \in \mathcal{FMG}_{\mathcal{K}}$ ) do
4     Foreach ( $g_1 \in g.\text{direct-subsets}$ ) do
5        $\mathcal{R} = \text{REPRESENTATIVE}(g_1)$ ;
6       Foreach ( $g_2 \in \mathcal{R}.\text{immediate-succs}$ ) do
7         If ( $g.\text{support} = g_2.\text{support} = \text{Supp}(g \cup g_2)$ ) then
8            $\text{MANAGE-EQUIV-CLASS}(g, g_2)$ ; /* $g, g_2 \in \mathcal{C}_g$  and  $g_2$  is the
9             representative of  $\mathcal{C}_g^*$ */
10          Else if ( $g.\text{support} < g_2.\text{support}$  and  $g.\text{support} =$ 
11             $\text{Supp}(g \cup g_2)$ ) then
12               $g$  is compared with  $g_2.\text{immediate-succs}$ ;
13              /*For the remainder of the element of
14                 $\mathcal{R}.\text{immediate-succs}$ ,  $g$  is compared only with each  $g_3$  |
15                 $g_3.\text{support} > g.\text{support}$ ;*/
16          If ( $\forall g_2 \in \mathcal{R}.\text{immediate-succs}$ ,  $\mathcal{C}_g$  and  $\mathcal{C}_{g_2}$  are incomparable)
17            then
18               $\mathcal{R}.\text{immediate-succs} = \mathcal{R}.\text{immediate-succs} \cup \{g\}$ ;
19 End

```

Algorithm 1: GEN-ORDER

same manner, PRINCE treats higher levels of the *minimal generator lattice* until reaching the maximal equivalence class(es). Due to lack of space, the pseudo-code of this step is omitted.

*Example 1.* Let us consider the extraction context  $\mathcal{K}$  given by Figure 1 (Left) for  $\text{minsup}=2$  and  $\text{minconf}=0.5$ . The first step allows the determination of the sorted set  $\mathcal{FMG}_{\mathcal{K}}$  and the negative border of minimal generators  $\mathcal{GBd}^-$ .  $\mathcal{FMG}_{\mathcal{K}} = \{(\emptyset, 5), (B, 4), (C, 4), (E, 4), (A, 3), (BC, 3), (CE, 3), (AB, 2), (AE, 2)\}$  and  $\mathcal{GBd}^- = \{(D, 1)\}$ . During the second step, PRINCE processes the element of  $\mathcal{FMG}_{\mathcal{K}}$  by comparing each frequent minimal generator  $g$ , of size  $k$  ( $k \geq 1$ ), with the lists of the immediate successors of its subsets of size  $(k-1)$ . Since the list of immediate successors of the empty set is empty, B is added to  $\emptyset.\text{immediate-succs}$ . Then, C is compared to B. BC is a minimal generator,  $\mathcal{C}_B$  and  $\mathcal{C}_C$  are thus incomparable and C is added to  $\emptyset.\text{immediate-succs}$ . E is then compared to this list. By comparing E to B,  $E.\text{support} = B.\text{support} = \text{Supp}(BE)$  and  $E \in \mathcal{C}_B$  whose B is the representative one. The MANAGE-EQUIV-CLASS function is then applied by replacing the occurrences of E, in the immediate successor lists, by B (in this case, there is no occurrence) and by continuing comparisons with B instead of E (in this case, there are no more comparisons to make with E). At this moment of processing, we have  $\emptyset.\text{immediate-succs} =$

$\{B, C\}$ . A is compared to B. As  $AB \in \mathcal{FMG}_K$ ,  $\mathcal{C}_B$  and  $\mathcal{C}_A$  are incomparable. On the other hand, by comparing A with C,  $A.\text{support} < C.\text{support}$  and  $A.\text{support} = \text{Supp}(AC)$  and then  $\mathcal{C}_A$  is a successor of  $\mathcal{C}_C$ . A is added to  $C.\text{immediate-succs}$  without any comparisons since it is still empty. BC is compared to the immediate successor lists of B and of C. Since, the list associated to B is empty, then BC is added. C has A as an immediate successor. A is then compared to BC and as  $BC.\text{support} = A.\text{support}$  but  $BC.\text{support} \neq \text{Supp}(ABC)$ ,  $\mathcal{C}_{BC}$  and  $\mathcal{C}_A$  are incomparable and BC is then added to  $C.\text{immediate-succs}$ . CE is compared to the immediate successor lists of C and of E. The list associated to C contains A and BC.  $\mathcal{C}_{CE}$  and  $\mathcal{C}_A$  are then incomparable since  $CE.\text{support} = A.\text{support}$  but  $CE.\text{support} \neq \text{Supp}(ACE)$ . By comparing CE to BC, since  $CE.\text{support} = BC.\text{support} = \text{Supp}(BCE)$  then CE will be affected to the equivalence class of BC. The `MANAGE-EQUIV-CLASS` function is then applied. In particular, comparisons of CE to the immediate successors of  $\mathcal{C}_E$  will be made with BC. As  $\mathcal{C}_E$  has B as a representative itemset, BC is compared to the elements of  $B.\text{immediate-succs}$ . However,  $B.\text{immediate-succs}$  contains only BC and the comparisons stop. It is the same for AB and AE. Having the *minimal generator lattice* built, an ascending sweeping is carried out from the  $\mathcal{C}_\emptyset$ . As  $(\emptyset)'' = \emptyset$ , there is no informative exact rule related to  $\mathcal{C}_\emptyset$ .  $\emptyset.\text{immediate-succs} = \{B, C\}$ . The frequent closed itemset associated to  $\mathcal{C}_B$  is then found and is equal to BE. The informative approximative rule  $\emptyset \Rightarrow BE$ , of a support equal to 4 and a confidence equal to 0.8, will be extracted. It is the same for  $\mathcal{C}_C$ . Using the same process and from  $\mathcal{C}_B$  and  $\mathcal{C}_C$ , the traversal of the *minimal generator lattice* is performed in an ascending way until extracting all the valid informative association rules. The resulting informative generic rules are sketched by Figure 1 (Right).

## 5 Experimental Results

In this section, we shed light on PRINCE performances vs those of CLOSE, A-CLOSE and TITANIC algorithms. PRINCE was implemented in the C language using gcc version 3.3.1. All experiments were carried out on a PC with a 2.4 GHz Pentium IV and 512 MB of main memory (with 2 GB of Swap) and running SuSE Linux 9.0.

In all our experiments, all times reported are real times, including system and user times. Figure 2 shows execution times of PRINCE<sup>(8)</sup> algorithm compared to those of CLOSE, A-CLOSE and TITANIC algorithms<sup>(9)</sup>.

- **Connect:** For this dense dataset, PRINCE performances are better than those of CLOSE, A-CLOSE and TITANIC for all *minsup* values. The Connect dataset is characterized by a great number of highly sized transactions. These characteristics complicate the task of CLOSE and A-CLOSE which have to carry

<sup>8</sup> For PRINCE algorithm, the value of *minconf* is set to 0.

<sup>9</sup> The authors are grateful to Yves Bastide who kindly provided source codes of CLOSE, A-CLOSE and TITANIC algorithms.

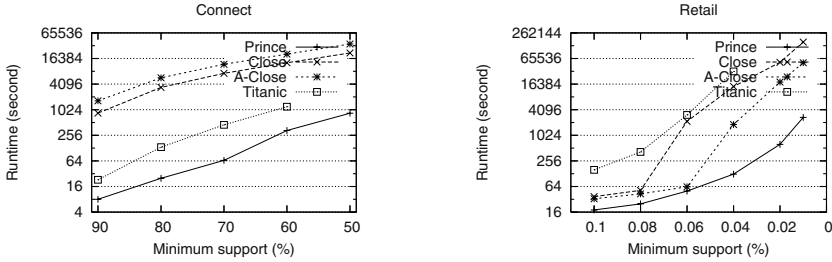


Fig. 2. PRINCE performances vs those of CLOSE, A-CLOSE and TITANIC

out expensive intersection computations. PRINCE and TITANIC avoid these expensive computations with an advantage for PRINCE. Indeed, PRINCE is favoured compared to TITANIC by a reduced cost of carried out comparisons for a frequent minimal generator compared with extension attempts carried out by TITANIC. Interestingly enough, for  $minsup = 50\%$ , the execution TITANIC could not come to an end for lack of memory space.

- **Retail:** For this sparse dataset, execution times of our algorithm are much more reduced than those of CLOSE, A-CLOSE, TITANIC algorithms. These performances can be explained by the enormous influence of the high number of items in the Retail dataset. Indeed, CLOSE is handicapped by a great number of candidates for which it is obliged to calculate closures, even though a high number of them is infrequent. The number of candidates affects also A-CLOSE performances because of the additional sweeping for each candidate as well as the cost of the closure computation step. On its side, TITANIC is considerably penalized by a great number of frequent items to consider in closure computations (for  $minsup = 0.04\%$ , 4643 items are frequent and the maximum size of a frequent minimal generator is only equal to 6 items). The execution of TITANIC stops, for support values lower than 0.004%, for lack of memory capacity. PRINCE performs better since the treatment to do for each frequent minimal generator is by far less expensive than that performed in the other algorithms.

## 6 Conclusion

In this paper, we proposed a new algorithm, called PRINCE, for an efficient extraction of frequent closed itemsets and their respective minimal generators as well as the generic rule bases. To this end, PRINCE builds the partial order contrary to the existing algorithms. A main characteristic of PRINCE algorithm is that it relies only on minimal generators to build the underlying partial order. Carried out experiments outlined that PRINCE largely outperforms existing "Test-and-generate" algorithms of the literature for both dense and sparse contexts. In the near future, we plan to add other constraints [16] to PRINCE algorithm so that the number of generic rules will be reduced while keeping the most interesting rules for the user.



## References

1. Goethals, B., Zaki, M.J.: FIMI'03: Workshop on frequent itemset mining implementations. In: FIMI Repository: <http://fimi.cs.helsinki.fi/>. (2003)
2. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. *Journal of Information Systems* **24** (1999) 25–46
3. Wille, R.: Restructuring lattices theory: An approach based on hierarchies of concepts. I. Rival, editor, *Ordered Sets*, Dordrecht-Boston (1982) 445–470
4. BenYahia, S., Nguifo, E.M.: Approches d'extraction de règles d'association basées sur la correspondance de Galois. In Boulicault, J.F., Cremilleux, B., eds.: *Revue d'Ingénierie des Systèmes d'Information (ISI)*, Hermès-Lavoisier. Volume 3–4. (2004) 23–55
5. Pasquier, N., Bastide, Y., Touil, R., Lakhal, L.: Discovering frequent closed itemsets. In Beeri, C., Buneman, P., eds.: *Proceedings of 7th International Conference on Database Theory (ICDT'99)*, LNCS, volume 1540, Springer-Verlag, Jerusalem, Israel. (1999) 398–416
6. Pei, J., Han, J., Mao, R., Nishio, S., Tang, S., Yang, D.: CLOSET: An efficient algorithm for mining frequent closed itemsets. In: *Proceedings of the ACM-SIGMOD DMKD'00*, Dallas, Texas, USA. (2000) 21–30
7. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing iceberg concept lattices with TITANIC. *Journal on Knowledge and Data Engineering (KDE)* **2** (2002) 189–222
8. Zaki, M.J., Hsiao, C.J.: CHARM: An efficient algorithm for closed itemset mining. In: *Proceedings of the 2nd SIAM International Conference on Data Mining*, Arlington, Virginia, USA. (2002) 34–43
9. Ganter, B., Wille, R.: *Formal Concept Analysis*. Springer-Verlag (1999)
10. Bastide, Y., Pasquier, N., Taouil, R., Lakhal, L., Stumme, G.: Mining minimal non-redundant association rules using frequent closed itemsets. In: *Proceedings of the International Conference DOOD'2000*, LNAI, volume 1861, Springer-Verlag, London, UK. (2000) 972–986
11. BenYahia, S., Cherif, C.L., Mineau, G., Jaoua, A.: Découverte des règles associatives non redondantes : application aux corpus textuels. *Revue d'Intelligence Artificielle (special issue of Intl. Conference of Journées francophones d'Extraction et Gestion des Connaissances (EGC'2003))*, Lyon, France **17** (2003) 131–143
12. Kryszkiewicz, M.: Concise representations of association rules. In Hand, D.J., Adams, N., Bolton, R., eds.: *Proceedings of Exploratory Workshop on Pattern Detection and Discovery in Data Mining (ESF)*, 2002, LNAI, volume 2447, Springer-Verlag, London, UK. (2002) 92–109
13. Kryszkiewicz, M.: Concise representation of frequent patterns based on disjunction-free generators. In: *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM)*, San Jose, California, USA. (2001) 305–312
14. Davey, B., Priestley, H.: *Introduction to Lattices and Order*. Cambridge University Press (2002)
15. Hamrouni, T., BenYahia, S., Slimani, Y.: PRINCE : Extraction optimisée des bases gnriques de règles sans calcul de fermetures. In: *Proceedings of the International Conference INFORSID* , Inforsid Editions, Grenoble, France. (2005) 353–368
16. Bonchi, F., Lucchese, C.: On closed constrained frequent pattern mining. In: *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, Brighton, UK. (2004) 35–42

# Efficient Compression of Text Attributes of Data Warehouse Dimensions

Jorge Vieira<sup>1</sup>, Jorge Bernardino<sup>2</sup>, and Henrique Madeira<sup>3</sup>

<sup>1</sup>Critical Software S.A.

`jvieira@criticalsoftware.com`

<sup>2</sup>CISUC-ISEC, Instituto Politécnico de Coimbra

`jorge@isec.pt`

<sup>3</sup>CISUC-DEI, Universidade de Coimbra

`henrique@dei.uc.pt`

**Abstract.** This paper proposes the compression of data in Relational Database Management Systems (RDBMS) using existing text compression algorithms. Although the technique proposed is general, we believe it is particularly advantageous for the compression of medium size and large dimension tables in data warehouses. In fact, dimensions usually have a high number of text attributes and a reduction in their size has a big impact in the execution time of queries that join dimensions with fact tables. In general, the high complexity and long execution time of most data warehouse queries make the compression of dimension text attributes (and possible text attributes that may exist in the fact table, such as false facts) an effective approach to speed up query response time. The proposed approach has been evaluated using the well-known TPC-H benchmark and the results show that speed improvements greater than 40% can be achieved for most of the queries.

## 1 Introduction

Over the last decades, the volume of data stored in databases has grown at a very high pace. Despite the fact that the evolution of storage capacity has followed this growth, a similar increase of disk access performance has not happened. On the other hand, improvements in speed of RAM memories and CPUs have outpaced improvements in physical storage devices by orders of magnitude. This technological trend led to the use of data compression, trading some execution overhead (to compress and decompress data) for the reduction of space occupied by data.

In a compressed database data is stored in compressed format on disk and is either decompressed immediately when read from disk or during query processing. In databases, and particularly in data warehouses, the reduction in the size of the data obtained by compression represents normally a gain in speed, as the extra cost in execution time (to compress and decompress the data) is compensated by the reduction in size of the data that have to be read/stored in the disks.

Data compression in data warehouses is particularly interesting for two main reasons: **1)** the amount of data normally stored in the data warehouse is very large,

potentially allowing considerable gains in data size, and 2) the data warehouses are used for querying only (i.e., only read accesses, as the data warehouse updates are done offline), which means that compression overhead is not relevant. Furthermore, if data is compressed using techniques that allow searching over the compressed data, then the gains in performance could be quite significant, as the decompression operation are only done when is strictly necessary.

In spite of the potential advantages of compression in databases, most of the commercial relational database management systems (DBMS) either do not have compression or just provide data compression at the physical layer (i.e., database blocks), which is not flexible enough to become a real advantage. Flexibility in database compression is essential, as the data that could be advantageously compressed is frequently mixed in the same table with data whose compression is not particularly helpful. Nonetheless, recent work on attribute-level compression methods has shown that compression can improve the performance of database systems in read-intensive environments such as data warehouses [1, 2].

Data compression and data coding techniques transform a given set of data into a new set of data containing the same information, but occupying less space than the original data (ideally, the minimum space possible). Data compression is heavily used in data transmission and data storage. In fact, reducing the amount of data to be transmitted (or stored) is equivalent to the increase of the bandwidth of the transmission channel (or the size of the storage device).

The first data compression proposals appeared in the 40's, namely proposed by D. Huffman, but these earlier proposals have evolved dramatically since then. In [3] is presented a survey of data compression covering all the period from the first proposals until the end of the 80's. More recent proposals can be found in [4].

The main emphasis of previous work has been on the compression of numerical attributes, where coding techniques have been employed to reduce the length of integers, floating point numbers, and dates [1, 5]. However, string attributes (i.e., attributes of type CHAR(n) or VARCHAR(n) in SQL) often comprise a large portion of database records and thus have significant impact on query performance.

In this paper we propose a flexible compression approach that allows the compression of data in Relational Database Management Systems (RDBMS) using existing text compression algorithms. This solution is specially appropriated for the compression of large dimension tables in data warehouses, as these tables usually have a high number of text attributes. The high complexity and long execution time of most data warehouse queries make the compression of dimension text attributes an effective approach to speed up query response time as shown by the experimental results obtained using TPC-H benchmark, where speed improvements greater than 40% have been observed.

The structure of the paper is as follows. Section 2 briefly summarizes the state of the art of data compression and coding techniques. Section 3 describes the proposed technique and section 4 evaluates experimentally the gain in space and performance obtained with the proposed technique. Section 5 concludes the paper.

## 2 Related Work

Data compression has been a very popular topic in the research literature and there is a large amount of work on this subject. The most obvious reason to consider compression in a database context is to reduce the space required in the disk. However, a maybe more important issue is whether the processing time of queries can be improved by reducing the amount of data that needs to be read from disk using a compression technique. There has been much work on compressing database indexes [6, 7] but less on compressing the data itself. With respect to compression of non-index data, traditional techniques such as Huffman coding [8] and Lempel-Ziv [9] work well for compressing certain types of data, such as medical images [10] or text [11], but are not advantageous to compress string fields in a database due to high execution costs. There are other algorithms for compressing numeric data [2, 12]. However, despite the abundance of string-valued attributes in databases, most existing work has focused on compressing numerical attributes [2, 6, 7, 12].

Recently, there has been a revival of interest on employing compression techniques to improve performance in a database. The data compression currently exists in the main databases engines, being adopted different approaches in each one of them.

Data compression in Oracle [13] is based on an algorithm specifically designed for relational databases that uses compression at data block level. In each block is created a table of symbols with correspondence to a dictionary of the attributes that are the compression target. The attributes are replaced in the block by pointers (links) for the table of symbols. The compression is only achieved in complete columns, however it can be used compression between columns (use the same value of the dictionary for different columns of the block) or of sequences of columns, when such could be advantageous. However, the use of this type of compression has a significant impact in data loading time and update time that increase significantly. To optimize these operations a table can be divided in temporal partitions and compressing only a partition when all estimate updates in this partition are done.

Teradata [14] presents a compression algorithm at the attribute level where compression candidates are all the attributes of fixed size that are not part of the existing index on the primary key of the table, with special advantage for columns with low cardinality. The compressed values are stored in a dictionary in the table header. This dictionary is constructed when tables are created or columns are added to an existing table, being the values to compress indicated by the user. The main advantage of this type of compression is the reduction of occurrences of decompression of the values, as decompression is only done when the compressed attributes are necessary for the construction of the query result (lazily decompressed).

IBM DB2 [15] is based on Lempel-Ziv compression algorithm, to make compression of lines. The compression algorithm is adapted accordingly to the data to compress, being the analysis of the data carried on samples. The compression can be made on all the tablespace or only on a partition. Each page has an anchor that indicates if that page is compressed or not. In the same way, each line has a bit that indicates if it is compressed or not. All the lines that are necessary for the execution of a query must be decompressed. In order to optimize the compression and decompression of data, a processor of compression by hardware can be used.

Sybase IQ [16] stores the data in a different form of the usual RDBMS. The data of one table is stored by columns instead of the traditional storage per line. The data

is indexed in arrays of bits for each column called Bit-Wise indexes. The index can contain all the distinct values if the values are just a few (less than 1000), or slices of values for columns with many distinct values. The particular form of storage of data is in itself the technique of compression used in Sybase IQ. This compression is obtained by two forms: elimination of the repeated values and reduction of the space occupied by the indexes. With Sybase IQ we only access data columns needed for the query. The data loading operations become heavier due to different form to store the data, however do not exist performance loss due to compression of the data.

The compression of data in databases offers two main advantages: less space occupied by data and potentially better query response time. If the benefit in terms storage is easily understandable, the gain in performance is not so obvious. This gain is due to the fact that less data had to be read of the storage, which is clearly the most time-consuming operation during the query processing.

The most interesting use of data compression and codification techniques in databases is surely in data warehouses, given the huge amount of data normally involved and its clear orientation for the query processing. As in the data warehouses all the insertions and updates are done during the update window [17], when the data warehouse is not available for users, off-line compression algorithms are more adequate, as the gain in query response time usually compensates the extra costs to codify the data before being loaded into the data warehouse. In fact, off-line compression algorithms optimize the decompression time, which normally implies more costs in the compression process. The technique presented in this paper follow these ideas, as it takes advantage of the specific features of data warehouses to optimize the use of traditional text compression techniques.

### 3 Attribute Compression

As mentioned before, some of the major commercial databases already offer compression mechanisms. However this compression features are generic and, consequently, not optimized to any specific scenario. Our approach is especially designed for data warehouses, particularly to compress dimension tables, which are usually composed by a large number of textual attributes with low cardinality. This approach also has the benefit of being independent from the RDBMS used. Therefore, using our approach we can implement data compression in databases that do not offer this option, such as Microsoft SQL Server or PostgreSQL. The use of this technique can also be combined with compression techniques already existing in some database engines, as the compression offered by Oracle 9iR2 and 10g.

The main objective of this technique is to allow the reduction of the space occupied by dimension tables with high number of rows, reducing the total space occupied by the data warehouse and leading to a consequent gains on performance. In fact, by reducing the size of large dimensions the star model becomes closer to the ideal star model proposed by R. Kimbal [17], with a consequent speed improvement.

The proposed approach aims to compress two different types of database attributes:

- Text attributes with low cardinality (referred further on as categories).
- Attributes of free text (comments or notes) that are mainly used for visualization (referred further on as descriptions).

Categories are textual attributes with low cardinality. Examples of category attributes are: city, country, type of product, etc. The codification of categories is made using 1 or 2 bytes, depending on the category cardinality. For categories with less than 256 distinct values the use of 1 byte is enough, the use of 2 bytes allows to codify categories with a maximum of 65791 (65535+256) distinct values, where the 256 most used are represented by an one byte code and the remaining less used values are represented by a two bytes code.

A description is a text attribute that is mainly used for visualization. In other words, it is not usually used to restrict, to group, or to order a query. An example of a description attribute is the comment attribute, which is frequently found in dimension tables. This type of attributes has the particularity of having a low access frequency and it is only necessary to decode it when the final result is being constructed. As this is a free text attribute its cardinality tends to be very high (attribute values are normally different one from each other), therefore using a codification similar to the one used for categories would result in an increase of the space occupied by the data. Therefore, for this type of attributes we propose the use of searchable text compression algorithms, which allow reducing the size of this type of attributes maintaining the ability of querying using this attributes to restrict the results. It is worth nothing that the codification of attributes does not imply any modifications at the physical structure of the database. The only change that may have to be made is the transformation of the CHAR attributes into VARCHAR attributes.

In order to guarantee that the data analysis applications (On-Line Analytical Processing - OLAP tools) continue to work transparently we propose the implementation a middleware to handle the compression and decompression of the attributes (using an approach similar to the one we have used in [18, 19]). That is, this middleware will perform query rewriting based on the metadata about the compressed attributes. The query originally received from the OLAP tool is intersected by the middleware and the values corresponding to compressed attributes are compressed before submitted the (modified) query to the DBMS. The query is executed normally and the result is decompressed only in the cases where compressed dimension attributes appear in the Select statement in the query (this will be detailed further on). This means that from the database point of view no changes are required as the data warehouse store searchable compressed attributes and the DBMS receives queries from the middleware with the values of attributed that are stored in a compressed form also compressed, which assure that the query is executed correctly.

### 3.1 Categories Coding

Categories coding is done through the following steps:

1. The data in the attribute is analyzed and a frequency histogram is build.
2. The table of codes is build based on the frequency histogram: the most frequent values are encoded with a one byte code; the least frequent values are coded using a two bytes code. In principle, two bytes are enough, but a third byte could be used if needed.
3. The codes table and necessary metadata is written to the database.
4. The attribute is updated, replacing the original values by the corresponding codes (the compressed values).

Table 1 presents an example of typical attributes of a client dimension in a data warehouse, which may be a large dimension in many businesses (e.g., e-business). For example, we can find several attributes that are candidates to coding, such as: CUST\_FIRST\_NAME, CUST\_LAST\_NAME, CUST\_MARITAL\_STATUS, CUST\_POSTAL\_CODE and CUST\_CITY.

Assuming that we want to code the CUST\_CITY attribute, an example of possible resulting codes table is shown in Table 2. The codes are represented in binary to better understand the idea. As the attribute has more than 256 distinct values, we will have codes of one byte to represent the 256 most frequent values (e.g. Berlin and Copenhagen) and codes of two bytes to represent the least frequent values (e.g. Dublin and Oporto). The values shown in Table 2 (represented in binary) would be the ones stored in the database, instead of the larger values. For example, instead of storing “Copenhagen”, which corresponds to 10 ASCII chars, we just stores one byte with the binary code “00000111”.

**Table 1.** Example of typical customer attributes and cardinality

Attribute	Type	Cardinality
CUST_ID	NUMBER	100000
CUST_FIRST_NAME	VARCHAR2(20)	800
CUST_LAST_NAME	VARCHAR2(40)	640
CUST_GENDER	CHAR(1)	2
CUST_YEAR_OF_BIRTH	NUMBER(4)	74
CUST_MARITAL_STATUS	VARCHAR2(20)	5
CUST_STREET_ADDRESS	VARCHAR2(40)	99435
CUST_POSTAL_CODE	VARCHAR2(10)	623
CUST_CITY	VARCHAR2(30)	620
CUST_COMMENT	VARCHAR2(200)	95444

**Table 2.** Codes Table example

Values	Code (in binary)
Berlin	00000101
Copenhagen	00000111
Dublin	00000001 00000010
Lisbon	00001001
London	00001011
Oporto	00000001 00000011
...	...

### 3.2 Descriptions Coding

Descriptions coding is very similar to the categories coding with the major difference that in this case the value in the attribute is not regarded as a single value, but as a set

of values (an ASCII string). Any text compression algorithm can be used to perform this type of compression, provided that it allows approximated (i.e., using text wild card) and exact search without decompression. Some text compression formats, like the ones presented in [20] and [21], allow this type of compression.

The key point to understand the coding approach used is that the compression algorithm includes the construction of a codes table similar to the one used in categories coding. Table 3 presents an example of the comment attribute of a typical customer table. In order to compress this attribute we first have to merge the values of all rows into a single text value, and then apply the compression algorithm in order to obtain a codes table, similar to the one presented in Table 4. Putting all the values in the comment attribute in a virtual same text file is needed to facilitate the determination of the frequency of each word. Again, as in category coding, we represent the most frequent word by one byte and the less frequent words by two bytes. If needed, we use a third byte for the even less frequent words.

After obtaining the codes table we must apply it, compressing the values in the table. The final result will depend on the compression algorithm used.

**Table 3.** Description attribute example

CUST_ID	CUST_COMMENT
1	The amount of time is not enough for processing
2	A quiz should be sent to this client
3	The client address is not complete
4	This client usually buys items in more than one store
5	This client should be deleted
6	There is something wrong whit the name
...	...

**Table 4.** Description codes table example

Values	Code
The	00000101
On	00000111
client	00001001
Name	00001011
This	00000001
...	...

As this type of attributes, when they exist, tend to occupy a large part of the table total space, their compression allows in general an impressive reduction in the size of the target table, while keeping the ability to search in the compressed values.



### 3.3 Query Execution

As mentioned, when using this compression approach the queries must be executed through a middleware that performs query rewriting and data decompression when necessary. In order to optimize the tasks of query rewriting and decompression the middleware has the codes metadata tables loaded in memory. In fact, this work as a small dictionary to allow translation from uncompressed values to compressed values and vice-versa.

Query rewriting is necessary in queries where the coded attributes are used in the WHERE clause for filtering. In these queries the values used for filter the result must be replaced by the correspondent coded values. Following are some simple examples of the type of query rewriting needed.

**Example 1.** The value 'LONDON' is replaced by the corresponded code, fetched from the codes table, shown in Table 2.

Original Query	Modified Query
SELECT CUST_NAME FROM CUSTOMERS WHERE CUST_CITY = 'LONDON'	SELECT CUST_NAME FROM CUSTOMERS WHERE CUST_CITY = 00001011

Note that the code is represented in binary (1 byte).

**Example 2.** The value 'L%' is replaced by the set of codes that exist in the codes table of Table 2 and that verify the condition.

Original Query	Modified Query
SELECT CUST_NAME FROM CUSTOMERS WHERE CUST_CITY like 'L%'	SELECT CUST_NAME FROM CUSTOMERS WHERE CUST_CITY in (00001001, 00001011)

**Example 3.** The values 'the' and 'client' are replaced by the correspondent compressed values, fetched from the codes table, shown in Table 4.

Original Query	Modified Query
SELECT CUST_NAME FROM CUSTOMERS WHERE CUST_COMMENT like '%the%client%'	SELECT CUST_NAME FROM CUSTOMERS WHERE CUST_COMMENT like '%00000101%00001001%'

In queries where the coded attributes are not used for filtering, it's not necessary to perform query rewriting.

### 3.4 Decompression

The decompression of the attributes is only made when the coded attributes are in the query select list. In these cases the query is executed and after that the result set is processed in order to decompress the attributes that contain compressed values. As the typical data warehousing queries return small result sets the decompression time will represent a very small amount of the total query execution time.

## 4 Experimental Results

The goal of the experiments performed is to measure experimentally the gains in storage and performance obtained using the proposed technique. We have implemented a simplified version of the middleware needed to compress and decompress the data and we used the TPC-H [22] performance benchmark as experimental setup. The size of the database was 1 GB (scale factor 1 in the TPC-H scaling rules). After analyzing the TPC-H schema we compressed the biggest dimensions (Orders, Part and Customer) and the fact tables (Lineitem and Partsupp).

The experiments were divided into two phases. In the first phase only categories compression was used. In the second phase we used categories compression in conjunction with descriptions compression.

### 4.1 Categories Coding Results

The three biggest dimensions (Orders, Part and Customer) and the Lineitem fact table were compressed using categories compression. Lineitem table was compressed, although it is a fact table, because this table has two degenerated dimensions that are text attributes with very low cardinality.

Table 5 presents the storage gains obtained after compressing the tables using categories compression. An average compression gain of 24.5% was obtained.

**Table 5.** Categories compression gains

<i>Table</i>	<i>Initial Size (MB)</i>	<i>Categories Compression</i>	
		<i>Size (MB)</i>	<i>Gain (%)</i>
Lineitem	856	640	25.2%
Orders	192	152	20.8%
Part	31	17	45.1%
Customer	28	26	7.1%
Total	1107	835	24.5%

### 4.2 Descriptions Coding Results

In TPC-H schema all the tables have a textual field used for store comments. As we saw before this kind of data can be compressed using searchable text compression algorithms. As these fields are substantially large, they represent a huge percentage in the total size of the tables.

Table 7 presents the gains obtained in storage after applying the descriptions compression in the biggest tables. This compression was applied after the categories compression and has resulted in an average compression ratio of 39.9%.

The text compression was made using a simple algorithm where the 127 most frequent words are coded with a character in the range of 0 (0000000) to 127 (0111111) and the less frequent words are coded with two characters, the first one in the range of 128 (1000000) to 192 (1100000) and the second one in the range of 193 (11000001) to 255 (1111111). This algorithm is limited for the compression of

4223 different words ( $127 + 64 \cdot 64$ ) and enables the search of words within the compressed text. Obviously, it is easily extended to the compression of more than 4223 different words by using a third coding character.

**Table 6.** Categories and descriptions compression gains

<i>Table</i>	<i>Initial Size (MB)</i>	<i>Attributes Compression</i>		<i>Descriptions Compression</i>	
		<i>Size (MB)</i>	<i>Gain (%)</i>	<i>Size (MB)</i>	<i>Gain (%)</i>
Lineitem	856	640	25.2%	536	37.3%
Partsupp	136	136	0%	72	47%
Orders	192	152	20.8%	104	45.8%
Part	31	17	45.1%	15	51.6%
Customer	28	26	7.1%	19	32.1%
<b>Total</b>	<b>1243</b>	<b>971</b>	<b>21.8%</b>	<b>746</b>	<b>39.9%</b>

This simple algorithm was chosen for its easiness of implementation and it does not offer the best compression ratio possible. As it is not optimal, the speedup that may be obtained will be a conservative value. Other existing compression algorithms, like the ones presented in [20] and [21] can be used to optimize the compression ratio.

### 4.3 Performance Results

In order to evaluate the performance speedup obtained with the compression performed a subset of the TPC-H queries were executed with the following configurations:

1. No compression
2. Categories compression
3. Categories compression and descriptions compression

**Table 7.** Queries execution times in seconds

<b>Query</b>	<b>No compression</b>	<b>Categories</b>	<b>Categories+Descriptions</b>
Q1	47	38	30
Q2	46	34	25
Q3	50	38	28
Q4	39	28	20
Q5	52	40	31
Q6	49	35	28
Q7	52	43	28
Q8	40	31	24
Q9	80	60	46
Q10	52	34	33
<b>Total time</b>	<b>507</b>	<b>382</b>	<b>295</b>
<b>Speedup(%)</b>	<b>-</b>	<b>24,6%</b>	<b>41,8%</b>

Table 7 presents the execution time in seconds of each query in the three configurations. We do not show the SQL of the TPC-H queries used for space reasons (some queries are quite long) but all the queries can be found in [22]. The use of categories compression resulted in an average speedup of 24,6%. The use of categories compression and descriptions compression resulted in an average speedup of 41,8%.

As can be observed in the results shown in Figure 1, all the queries suffered a reduction in execution time when we use categories compression and the query execution time is considerable reduced when we use categories in conjunction with description compression.

## 5 Conclusions

This paper proposes and evaluates a general approach that allows the compression of data in RDBMS, which is particularly advantageous for the compression of medium size and large dimension tables in data warehouses. In fact, large dimensions usually have a high number of text attributes and a reduction in the size of middle or large dimension have a big impact in the execution time of queries that join that dimension with the fact tables. In general, the high complexity and long execution time of most data warehouse queries make the compression of dimension text attributes and possible text attributes that may exist in the fact table an effective approach to speed up query response time.

The proposed technique includes coding of two types of dimension attributes: categories and descriptions. The former are attributes with low cardinality (typically, text attributes) while the latter are attributes such as comments and descriptions (free text attributes). This approach also has the benefit of being independent from the RDBMS used. Therefore, using our approach we can implement data compression in databases that do not offer this option, or combined with compression techniques already existing in some database engines.

The proposed approach has been evaluated using the well-known TPC-H benchmark and the results have shown that it is possible to obtain a significant reduction of approximately 40% in the space occupied by TPC-H tables. The results also show a speedup improvement better than 40% for most of the queries.

## References

1. Goldstein, J., Ramakrishna, R., Shaft, U.: Squeezing the most out of relational database systems, In Proc. of ICDE (2000) page 81.
2. Westmann, T., Kossmann, D., Helmer, S., Moerkotte, G.: The Implementation and Performance of Compressed Databases, ACM SIGMOD Record Vol 29, No 3 (2000) 55-67
3. Lelewer, D., Hirschberg, D.: Data Compression, ACM Computing Surveys (1987)
4. Data Compression Conference. DCC Home page: <http://www.cs.brandeis.edu/~dcc/index.html>.
5. Roth, M., Horn, S.: Database compression, SIGMOD Record, 22(3):31-39 (1993)

6. Gray, J., Reuter. A.: *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann (1993)
7. Ramakrishnan, R., Gehrke, J.: *Database Management Systems*, McGraw Hill (2000)
8. D Huffman: A Method for the Construction of Minimum Redundancy Codes, *Proc IRE*, 40(9) (1952) 1098-1101
9. Ziv, J., Lempel, A.: A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory*, 22(1) (1997) 337-343.
10. Karadimitriou, K., Tyler, J.: Min-Max Compression Methods for Medical Image Databases, *SIGMOD Record*, Vol 26, No 1 (1997)
11. Moffatt, A., Zobel, J.: Text Compression for Dynamic Document Databases, *IEEE Transactions on Knowledge and Data Engineering*, Vol 9, No 2 (1997)
12. Chen, Z., Gehrke, J., Korn, F.: Query Optimization In Compressed Database Systems, *ACM SIGMOD* (2001) 271-282
13. Poess, M., Potapov, D.: Data Compression in Oracle, *Proceedings of the 29th VLDB Conference* (2003)
14. Morris, M.: *Teradata Multi-Value Compression V2R5.0*, Teradata White Paper (2002), available at <http://www.teradata.com/t/page/86995/>.
15. IBM Redbooks: *DB2 V3 Performance Topics* (1994) 75-90, available at <http://publib-boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/gg244284.html?Open>.
16. Krneta, P.: *Sybase Adaptive Server® IQ with Multiplex*, Sybase White Paper (2000), available at <http://www.sybase.com/products/databaseservers/asiq>.
17. Kimball, R., Ross, M.: *The data warehouse toolkit*, 2nd edition, Ed. John Wiley & Sons, Inc (2002)
18. Bernardino, J., Furtado, P., Madeira, H.: Approximate Query Answering Using Data Warehouse Striping, *Journal of Data and Knowledge Engineering*, Volume 19, Issue 2, Elsevier Science Publication (2002)
19. Costa, M., Vieira, J., Bernardino, J., Furtado, P., Madeira, H.: A middle layer for distributed data warehouses using the DWS-AQA technique, *8th Conference on Software Engineering and Databases* (2003)
20. Brisaboa, N., Iglesias, E., Navarro, G., Paramá, J.: An Efficient Compression Code for Text Databases, *ECIR* (2003) 468-481
21. Silva da Moura, E., Navarro, G., Ziviani, N., Baeza-Yates, R.: Compression: A Key for Next-Generation Text Retrieval Systems, *ACM transactions on informations systems* (2000) 113-139
22. TPC-H benchmark. <http://www.tpc.org/tpch/spec/tpch2.1.0.pdf>

# Effectiveness of Document Representation for Classification

Ding-Yi Chen, Xue Li, Zhao Yang Dong, and Xia Chen

School of Information Technology and Electrical Engineering,  
University of Queensland, QLD 4072, Australia  
{dingyi, xueli, zdong, chenxia}@itee.uq.edu.au

**Abstract.** Conventionally, document classification researches focus on improving the learning capabilities of classifiers. Nevertheless, according to our observation, the effectiveness of classification is limited by the suitability of document representation. Intuitively, the more features that are used in representation, the more comprehensive that documents are represented. However, if a representation contains too many irrelevant features, the classifier would suffer from not only the curse of high dimensionality, but also overfitting. To address this problem of suitability of document representations, we present a classifier-independent approach to measure the effectiveness of document representations. Our approach utilises a labelled document corpus to estimate the distribution of documents in the feature space. By looking through documents in this way, we can clearly identify the contributions made by different features toward the document classification. Some experiments have been performed to show how the effectiveness is evaluated. Our approach can be used as a tool to assist feature selection, dimensionality reduction and document classification.

## 1 Introduction

Instead of documents themselves, automatic classifiers learn from modelled documents, that is, the representation of original documents. However, the capability of a representation may limit the effectiveness of document classification. *Conflict instances* are the cases in training such that they have identical representations but are associated with different labels. This is also a case of incomplete document representation.

To address the representation of incompleteness, one might suggest to apply more sophisticated document representation. Nevertheless, several researches [1,2,3] found that the sophisticated representations do not significantly improve the effectiveness because of the overfitting problem. As Sebastiani defined in his work [4], overfitting is a problem that classifier wrongly puts too much weight on unimportant features.

Incomplete representation is caused by lacking of related features, while overfitting is derived from superfluous features. Either of them degrades classification

performance. Conventional methods to evaluate the suitability of document representations are averaging multiple effectiveness scores of various classifiers. Indeed, these scores do reflect the actual effectiveness of classifiers with particular document representations. However, in terms of representation evaluation, the scores derived from classifiers are limited by the learning capacity of a classifier.

We provide a classifier-independent document representation evaluation method, which utilises a labelled corpus as a sample of document distribution in the feature space. If there were no conflict instances in the feature space, the representation would be said as complete. The work [5] show the similar idea. However, the method in the work [5] needs to calculate every points in the feature space, which is infeasible for high dimensionality such as document classification, not to mention the lack of statistical significance in high dimensional space. In order to reduce the high dimensionality, we map the whole feature set into an essential feature subset, whose size is relatively small. Then we utilise the effectiveness evaluation techniques to estimate the effectiveness of representation on the labelled corpus, which is a collection of sample documents collection in a specified domain.

The benefits of our approach are: 1) it reduces the dimensionality of the problem space, which simplify the computation as well as diminish the possibility of overfitting; and 2) it provides a measurement for representations that have different sizes.

The paper is organised as follows: Section 2 lists influential related work. Section 3 explains our proposed approach to address the problem of incompleteness and overfitting of document model. Section 4 illustrates our experiment result. Section 5 provides our conclusion.

## 2 Influential Previous Studies

A classifier does not work on documents directly, instead, it works on the representation (i.e, features) of documents. The representation can be a set of words (binary independence retrieval (BIR), a.k.a. bag of words [6]), a vector with weighted words (vector space model (VSM) [7]), or the result of latent semantic analysis [8].

How do we compare these document representations? Lewis [3] uses the *Max-Cat* text categorization package [9] which is based on the probabilistic classifier to compare the word-based model with the phrase indexing model. Apte *et al* [1] applies the SWAP-1 [10] rule induction method to compare the effectiveness among subset of features. Dumais [2] uses Rocchio[11] decision trees [12], naive Bayes [13], Bayes net [14], and support vector machine [15] to test the result of feature selection. These methods rely on the learning capabilities of classifiers, which might suffer from under-fitting and overfitting.

The overfitting problem is usually referred as a generalisation problem in classification research [16]. The generalisation error of a classifier can be estimated by structural risk minimisation [17], Akaike Information Criterion [18], and Bayesian Information Criterion [19]. However, these methods examine the classifier model instead of document representation. For example, VC-dimension

analysis is not suitable for fitness of document representation, because the representation does not need to be shattered.

A conventional document representation usually has thousands of features, and many of which might be uninformative. Thus, we apply the feature selection techniques to choose the essential feature subset. There are two main approaches, wrappers and filters. The wrappers utilise classifiers such as the sub-optimal algorithms [20] or decision trees [21] to weight the importance of features; while filters compute the importance of features by feature significance measurements such as document frequency, information gain,  $\chi^2$  independence, mutual information, term strength [22], and term contribution [23]. In this paper, the filter approach is chosen for its classifier-independence and simplicity.

### 3 Problem Statement and Proposed Approach

Suppose a document representation uses the feature set  $F$  to represent documents, and the labelled document corpus  $\Phi : D \times C$  is a Cartesian product of the document set  $D$  and the category labels set  $C$ . The research problem addressed in this paper can be formulated as follow: Given an essential feature subset  $\hat{F} \subseteq F$ , a labelled corpus  $\Phi$ ; find the effectiveness  $\rho_{(F,\Phi)}$  of the document representation.

The term ‘*effectiveness*’ means the correctness of classification decisions. Similarly, *representation effectiveness* indicates the effectiveness of representation, which can estimates the correctness of classification decisions. Effectiveness is usually expressed as precision-recall break-even,  $F_\beta$  function, accuracy, or Pearson correlation. In this paper, we choose Pearson correlation as our measurement instrument. The detailed calculation is shown at formula 3.

An *essential feature subset (EFS)* serves as a snapshot of corresponding document representation The EFS is selected by feature signification measurement, whose detail is shown in section 3.3. The more significant the feature is, the higher priority it is selected to the EFS. In this paper, we assume the size of EFS is fixed, which reflects the circumstance in limited computation environments with a relatively small number of features. The details of effectiveness evaluation on a given feature set or subset is discussed in following subsections.

#### 3.1 Representing Documents in a Dimensional Space

Respecting the feature set  $F$ , a document  $d$  is decomposed as:

$$d \xrightarrow{F} [f_1, f_2, \dots, f_{|F|}], \quad (1)$$

where  $|F|$  is the size of  $F$ . With formula 1, a document can be considered as a point in  $|F|$ -dimensional space. We call the space *representation space*.

Due to the high dimensionality of the representation space, we map the points in the representation space to the space constructed by essential feature subset  $\hat{F}$ , namely, *model space*. A point in model space is called *bucket*, for it may hold multiple documents.



### 3.2 Measuring the Representation Effectiveness

The effectiveness of document representation indicates whether the feature set  $F$  can fully express a document. If not, no matter how good the classifier is, misclassification is not avoidable. We use the correctness of classification decisions, which is negatively related to the number of discovered conflict instances, to measure the representation effectiveness. There are four types of classification decisions:

1. True Positive (TP): Classifier correctly makes positive decisions.
2. True Negative (TN): Classifier correctly makes negative decisions.
3. False Positive (FP): Classifier mistakenly makes positive decisions.
4. False Negative (FN): Classifier mistakenly makes negative decisions.

Table 1 further shows the agreement between correct label and classifier decision [4].

**Table 1.** The contingency table of correct label and classifier decisions.

Should document $d$ be filed to $c$		Correct Label	
		Positive	Negative
Classifier	Positive	True Positive (TP)	False Positive (FP)
Decision	Negative	False Negative (FN)	True Negative (TN)

In some cases, the costs of FP and FN are not equal. For example, in email filtering, FP is severer than FN. Here, we utilise threshold  $\tau$  to reflect the costs of misclassification. According to [24], maximum effectiveness can be achieved if we set the threshold  $\tau$  as:

$$\tau = \frac{\lambda_{FP} - \lambda_{TN}}{(\lambda_{FN} - \lambda_{TP}) + (\lambda_{FP} - \lambda_{TN})}, \quad (2)$$

where  $\lambda_{TP}, \lambda_{FP}, \lambda_{TN}, \lambda_{FN}$  are the penalty for TP, FP, TN and FN.

Finally the representation effectiveness  $\rho$  can be computed as:

$$\rho = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(FP + TN)(FN + TN)}}, \quad (3)$$

where  $TP, TN, FP, FN$  are numbers of TP, TN, FP, and FN decisions. In this paper we use the Pearson Correlation as our effectiveness measurement, however, other effectiveness measurement such as  $F_\beta$  function (Formula 4) or accuracy (Formula 5) can also be used to measure to representation effectiveness.

$$F_\beta : \frac{(\beta^2 + 1)TP}{\beta^2(TP + FN) + (TP + FP)}, \quad (4)$$

$$\text{Accuracy} : \frac{TP + TN}{TP + TN + FP + FN}, \quad (5)$$

where  $\beta$  is a parameter which reflects relative degree of importance attributed to precision and recall. Usually parameter  $\beta = 0$ , which attributes equal importance of precision and recall. If it is not the case, set  $\beta < 1$  if precision is more important, otherwise set  $\beta > 1$  to emphasise recall [25].

### 3.3 Generating Essential Feature Subset

The EFS possess the most significant features for classification. Usually, the significance of feature is generated by feature significance function, such as  $\chi^2$  independence, information gain and mutual information. Yang et al. [22] found that document frequency,  $\chi^2$  independence score information gain provide similar good performance on feature selection. In this paper, we use  $\chi^2$  independence as feature signification measurement provided by Yang et al. [22], which is shown as Formula 6:

$$\chi^2(f, c) = \frac{N \times (AD - BC)^2}{(A + B)(A + C)(B + D)(C + D)} \quad (6)$$

where  $N$  is the total number of documents,  $A$  is the number of times  $f$  and  $c$  co-occur (act like TP if consider  $f$  is the only feature);  $B$  is the number of times the  $f$  occurs without  $c$  (FP);  $C$  is the number of times the  $c$  occurs without  $f$  (FN);  $D$  is the number of times neither  $f$  nor  $c$  occurs (TN). The larger the number, the more significant the feature is.

Given the size of EFS  $|\hat{F}|$ , the EFS can be obtained by selecting the most significant  $|\hat{F}|$  features. Since a feature may have different significance scores with each category, the overall significance of a feature  $\xi(f)$  is defined as the maximum significance score of the feature, that is:

$$\xi(f) = \max_{c \in C} \{\chi^2(f, c)\} \quad (7)$$

### 3.4 Estimating the Probability of Positive

The number of TP, TN, FP, and FN, can be estimated by the probability of positive (POP) of each bucket. For a given category  $c$ , POP of a bucket  $b$  can be calculated by:

$$POP_c(b) = \frac{\sum_{d \in b} \Phi_c(d)}{|b|} \quad (8)$$

where  $|b|$  is the number of instances in  $b$ , and  $\Phi_c(d)$  is the value of classification decision (1 for positive, 0 for negative) on the pair  $(d, c)$ ,  $d \in D$ ,  $c \in C$ .

In terms of decision making, if  $POP \geq \tau$ , then a positive decision is made. However, if  $POP$  is less than 1, then there will be  $1 - POP$  chance that the decision is FP. Similarly, if  $0 < POP < \tau$ , then the chance that the decision is FN is  $POP$ . We use the following formulae to obtain the probability of TP, TN, FP, and FN for category  $c$ :

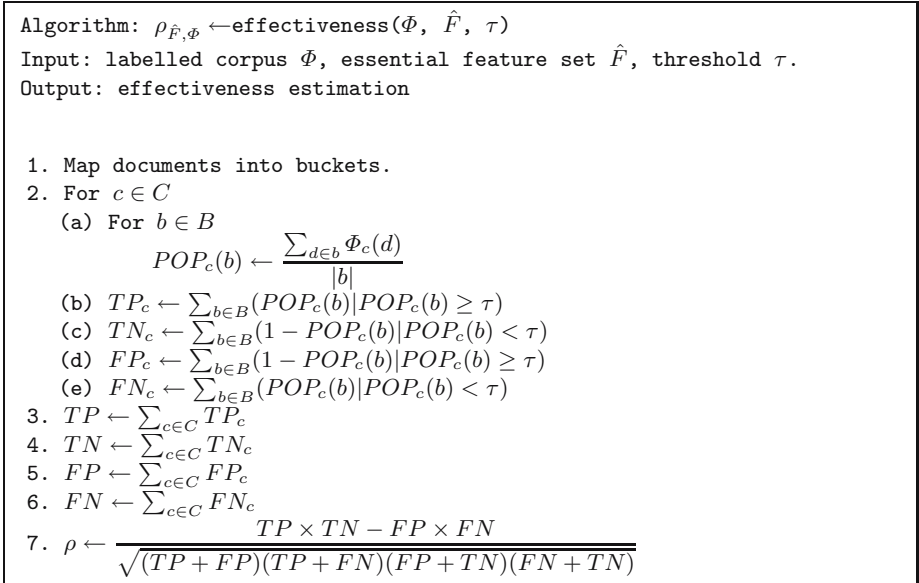
$$\begin{aligned}
TP_c &= \sum_{b \in B} (POP_c(b) | POP_c(b) \geq \tau) \\
TN_c &= \sum_{b \in B} (1 - POP_c(b) | POP_c(b) < \tau) \\
FP_c &= \sum_{b \in B} (1 - POP_c(b) | POP_c(b) \geq \tau) \\
FN_c &= \sum_{b \in B} (POP_c(b) | POP_c(b) < \tau)
\end{aligned} \tag{9}$$

where  $B$  is the set of non-empty buckets.

Finally, the numbers of TP, TN, FP and FN can be calculated as:

$$\begin{aligned}
TP &= \sum_{c \in C} TP_c, & TN &= \sum_{c \in C} TN_c \\
FP &= \sum_{c \in C} FP_c, & FN &= \sum_{c \in C} FN_c
\end{aligned} \tag{10}$$

With the numbers of TP, TN, FP and FN, the effectiveness of a document representation can be obtained by Formula 3, which is listed in section 3.2. The detail algorithm is shown in Figure 1.



**Fig. 1.** The representation effectiveness computing algorithm

The effectiveness of representation respecting different EFS is examined by the experiments in section 4.

## 4 Experiments and Results

This section compares the fitness of different document models upon the corpus Reuters-21578 [26]. The Reuters-21578 contains 21,450 valid documents and 135 categories. Each document may have zero to multiple category labels.

In this work, two document modelling methods: binary independence retrieval (BIR) and vector space model (VSM), combined with two stemmers: KStem [27] and Porter [28] have been examined with our method. Thus totally effectiveness of four representations: BIR-KStem, BIR-Porter, VSM-KStem, and VSM-Porter were put into investigations.

#### 4.1 Experiment Design

Before each experiments, the words in Moby most frequent 1,000 words list [29] and numbers were considered as non-informative words and removed.

For each representation, we selected 10 essential feature subsets which contain the most significant 50, 100, 150, . . . , 500 features, which were selected by the  $\chi^2$  statistic measure. In each trail, one essential feature subset was selected, together with the Apte training set of Reuters corpus and the threshold  $\tau = 0.5$ , were fed into the representation effectiveness computing algorithm shown in Figure 1).

The output of representation effectiveness computing algorithm in each trail was an estimation of effectiveness of corresponding representation.

#### 4.2 Experiment Results

The experiment results show the relationship between the size of essential feature subset and estimated effectiveness.

Table 2 and Figure 2 contain the result of representation effectiveness analysis. The table tells the effectiveness for each relationship between the size of essential feature subset and estimated effectiveness. and the figure shows the relationship between the size of essential feature subset and estimated effectiveness. According to the Table 2 and Figure 2, we can found that: firstly, the larger size of essential feature subset, the better the effectiveness of document representation be measured; secondly, with around 500 significant features, the effectiveness is able to reach 90%; and finally, the VSM is slightly better than BIR, and the Porter stemmer is also slightly better then KStem, which support the results of [1,2,3].

**Table 2.** Effectiveness comparisons between four document representations.

Essential Feature Subset (EFS) Size	Document Representation			
	BIR-KStem	BIR-Porter	VSM-KStem	VSM-Porter
50	0.608	0.610	0.615	0.611
100	0.635	0.645	0.649	0.652
150	0.662	0.681	0.683	0.692
200	0.689	0.717	0.728	0.746
250	0.724	0.753	0.786	0.802
300	0.768	0.790	0.816	0.821
350	0.808	0.818	0.836	0.841
400	0.832	0.842	0.857	0.860
450	0.856	0.868	0.877	0.880
500	0.880	0.893	0.898	0.900

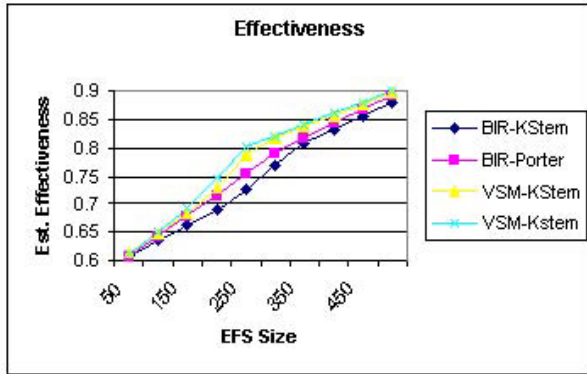


Fig. 2. Relative Effectiveness comparisons

## 5 Conclusions

Due to the immense cost of computing high dimensionality such as that in document classification, it is important to be able to measure the effectiveness of document representation. In this paper, we designed a heuristic, classifier-independent method to measure the effectiveness of document representation for classification. In our experiments, our method not only can provide a solution for comparing and contrasting the effectiveness of different document representations, but also find that only a small percentage of features (e.g., 2.74%, 500 out of 18234) can reach the 90% effectiveness.

Our perception of features in terms of their ability in classifying documents has shown a new way of feature selection. The proposed approach can further benefit the design of document representations.

## Acknowledgments

The work reported in this paper was funded in part by the Australian Research Council - Discovery Project Grant (ARC DP0558879), and computational resources used in this work were provided by the Queensland Parallel Supercomputing Foundation (QPSF).

## References

1. Apte, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)* **12** (1994) 233–251
2. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: *Proceedings of the seventh international conference on Information and knowledge management*, Bethesda, Maryland, United States, ACM Press (1998) 148–155

3. Lewis, D.D.: An evaluation of phrasal and clustered representations on a text categorization task. In: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, Copenhagen, Denmark, ACM Press (1992) 37–50
4. Sebastiani, F.: Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)* **34** (2002) 1–47
5. Chen, D.Y., Li, X., Dong, Z.Y., Chen, X.: Determining the fitness of a document model by using conflict instances. In: The Sixteenth Australasian Database Conference, Newcastle, Australia, Australian Computer Society Inc. (2005) 125–134
6. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. *Journal of the American Society for Information Science* **27** (1976) 129–146
7. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. *Communications of the ACM* **18** (1975) 613–620
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology* **41** (1990) 391–407
9. Lewis, D.D.: Representation and learning in information retrieval. Phd thesis, University of Massachusetts (1992)
10. Weiss, S.M., Indurkha, N.: Optimized rule induction. *IEEE Expert* **8** (1993) 61–69 TY - JOUR.
11. Rocchio, J.: Relevance feedback in information retrieval. In Salton, G., ed.: *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall (1971) 313–323
12. Chickering, D.M., Heckerman, D., Meek, C.: A Bayesian approach to learning Bayesian networks with local structure. In: Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann (1997) 80–89
13. Lewis, D.D.: Naive (bayes) at forty: The independence assumption in information retrieval. In: Proceedings of ECML-98, 10th European Conference on Machine Learning, Chemnitz, DE, Springer Verlag, Heidelberg, DE (1998) 4–15
14. Heckerman, D., Geiger, D., Chickering, D.M.: Learning bayesian networks: The combination of knowledge and statistical data. In: KDD Workshop. (1994) 85–96
15. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In Rouveirol, C.N., Celine, eds.: Proceedings of ECML-98, 10th European Conference on Machine Learning, Chemnitz, DE, Springer Verlag, Heidelberg, DE (1998) 137–142
16. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines: and other kernel-based learning methods. Cambridge : Cambridge University Press (2000)
17. Vapnik, V.N.: Constructing learning algorithm. In: *The Nature of Statistical Learning Theory*. Springer-Verlag New York, New York, NY (1995) 119–156
18. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In Petrov, B.N., Csaki, F., eds.: *Second International Symposium on Information Theory, Armenia* (1974) 267–281
19. Schwarz, G.: Estimating the dimension of a model. *Annals of Statistics* **6** (1978) 461–464
20. Dietterich, T.: Overfitting and undercomputing in machine learning. *ACM Computer Survery* **27** (1995) 326–327
21. Quinlan, J.R., Cameron-Jones, R.M.: Oversearching and layered search in empirical learning. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Canada, Morgan Kaufmann Publishers (1995) 1019–1024

22. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In Fisher, D.H., ed.: Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, US, Morgan Kaufmann Publishers, San Francisco, US (1997) 412–420
23. Liu, T., Liu, S., Chen, Z., Ma, W.Y.: An evaluation on feature selection for text clustering. In Fawcett, T., Mishra, N., eds.: ICML 2003: The 20th International Conference on Machine Learning, AAAI Press (2003) 488–495
24. Lewis, D.D.: Evaluating and optimizing autonomous text classification systems. In: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, United States, ACM Press (1995) 246–254
25. Van Rijsbergen, C.J.: Evaluation. In: Dept. of Computer Science, University of Glasgow. Department of Computer Science, University of Glasgow (1979)
26. Lewis, D.D.: Reuters corpus (21578).  
<http://www.daviddlewis.com/resources/testcollections/reuters21578/> (2000)
27. Krovetz, R.: Viewing morphology as an inference process. In: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 93), Pittsburgh, Pennsylvania, United States, ACM Press (1993) 191–202
28. Porter, M.F.: An algorithm for suffix stripping. In Sharp, H.S., ed.: Readings in Information Retrieval. Morgan Kaufmann, San Francisco (1997)
29. Ward, G.: Moby Word–Moby lexicon project.  
<http://www.dcs.shef.ac.uk/research/ilash/Moby/mwords.html> (1996)

# 2-PS Based Associative Text Classification

Tieyun Qian, Yuanzhen Wang, Hao Long, and Jianlin Feng

Department of Computer Science,  
Huazhong University of Science and Technology,  
Wuhan 430074, Hubei, China  
tieyun\_qian@yahoo.com.cn, wyzh1999@371.net, lonho@163.net,  
fengjl@mail.hust.edu.cn

**Abstract.** Recent studies reveal that associative classification can achieve higher accuracy than traditional approaches. The main drawback of this approach is that it generates a huge number of rules, which makes it difficult to select a subset of rules for accurate classification. In this study, we propose a novel association-based approach especially suitable for text classification. The approach first builds a classifier through a 2-PS (Two-Phase) method. The first phase aims for pruning rules locally, i.e., rules mined within every category are pruned by a sentence-level constraint, and this makes the rules more semantically correlated and less redundant. In the second phase, all the remaining rules are compared and selected with a global view, i.e., training examples from different categories are merged together to evaluate these rules. Moreover, when labeling a new document, the multiple sentence-level appearances of a rule are taken into account. Experimental results on the well-known text corpora show that our method can achieve higher accuracy than many well-known methods. In addition, the performance study shows that our method is quite efficient in comparison with other classification methods.

## 1 Introduction

The application of association rules in solving the classification problem was first introduced in [1]. Since then, associative classification has received a lot of attentions. Researchers in automatic text categorization also borrow this idea for its high accuracy.

Through the exhaustive search in data, all rules meeting user-specified minimum support and confidence will be found. This is the main strength of associative classification because more accurate classifier can be derived from the rules that provide a general description of the data. Along with the strength, a main drawback in association classification is that it's quite difficult to select useful rules from such a huge number of rules for accurate classification. Extensive research has been carried out to solve this problem. However, all these methods select rules only by machine learning or statistic means [2], [3], even the associative text categorization methods [4], [5] don't make use of the characteristics of a text document to prune rules.

In this paper, we first mine the frequent rules (frequent itemset with the class label as consequent) in each category at the document level. Then we present a novel 2-PS (Two-Phase) method that is designed especially for text categorization. In its local phase, what we emphasize on is to greatly reduce the number of rules using a



sentence-level support threshold. This pruning strategy is especially suitable for text classification because the basic semantic unit in a document is just a sentence. Through this local pruning step, most redundant and semantic unrelated rules are removed. In its global phase, all the remaining rules are compared and selected with a global view, i.e., training examples from different classes are merged together to evaluate rules. The evaluation process is performed based on a minimum confidence as well as a covering method. The rules whose confidences are lower than minimum confidence are removed. The covering method checks several rules with similar entropy simultaneously and chooses the rule having highest support, and hence avoids overfitting problem of small disjunction and speeds up the sequence covering procedure.

Moreover, the multiple occurrences of one rule are taken into account in prediction. We do this for two reasons: first, sentence-level rules should match sentences in a test document; second, several appearances of a rule means that it maybe the center of topic of this document.

The outline of this paper is arranged as follows: Section 2 reviews related work in associative classification and associative text categorization. In section 3, we introduce our 2-phase rule pruning and selection method. Section 4 discusses how to predict an unseen example using the rules found. The experimental results on real data sets are reported in Section 5 as well as the performance study. We conclude our study in section 6.

## 2 Related Works

Associative classification was first developed in [1], and from then on, it is gaining popularity for its high accuracy. Until now, research has been mainly focused on how to prune rules and then build an accurate classifier. As in traditional rule generation practice, the popularity and significance of a rule were firstly studied. Besides the pessimistic error rate in [1], other methods such as chi-square test were also used in [2], [6]. In order to construct an accurate classifier, [1], [2] modified the sequential covering method to determine when to stop adding more rules into the classifier. [7] employed the intensity of implication rather than the confidence to sort CARs and made use of ROC curve to judge the proper size of classifier.

Associative classification also has many applications in text classification. In [4], the data sets were stored in a transactional database with words as items and each document as a transaction. The algorithm in [5] represented data sets in the same manner. The main difference between these two papers is that the former constructs classifier directly from association rules while the latter constructs a Bayesian classifier using subsets of frequent itemsets to produce a probability estimate. A similar method was adopted to mine generalized association rules in [8] though two special ranking criteria were proposed for the hierarchical classification task. [9] mined the frequent itemsets of substructures within XML documents.

All the current associative text classification technologies have a common characteristic that they all exploit groups of co-occurring words in the same document. [9] is an exception, but it was developed for XML data. In our previous work [10], we viewed each word as an item and each sentence as a transaction, and then directly selected the most significant itemsets for constructing a category classifier. However, the rule pruning and evaluation method in that paper were different from those in this paper, and the classification approach was different either.

### 3 Two-Phase Based Classifier Building

Given a set of training documents, we construct the classifier through several steps as follows. In the first step, we use an apriori-based algorithm to mine all document-level frequent rules in each category. In next step, document-level frequent rules are pruned by sentence so that only sentence-level frequent rules are kept for further selection. In third step, training examples from all categories are merged together to calculate the confidence and the negative entropy value of a rule. More accurate rules are then selected according to minimum confidence threshold and database coverage. More details are presented in the subsections below.

#### 3.1 Local Pruning Phase

The itemsets contained in a document-level frequent rule are actually a set of words that often co-occur in the same document, and they always span several sentences. Since two words occurring in the same sentence seem more correlated than two words occurring far apart in a document [11], it's worthwhile to prune document-level

Algorithm: Local pruning

Input: document-level frequent rule set of category  $C$ ,  $DFRc$ ; a set of training documents of category  $C$ ,  $Tc$ ; a user-specified sentence-level minimum support,  $minSentSup$

Output: sentence-level frequent rule set of category  $C$ ,  $SFRc$

Method:

- (1)  $SFRc = \emptyset$
- (2) foreach document  $d$  in  $Tc$
- (3)      $R = \emptyset$
- (4)     foreach sentence  $s$  in  $d$
- (5)         foreach rule  $r$  in  $DFRc$
- (6)             if ( $r \subseteq s$ )
- (7)                  $R = R \cup \{r\}$ ,  $SFRc = SFRc \cup \{r\}$
- (8)     foreach rule  $r$  in  $SFRc$
- (9)         if ( $r \in R$ )  $r.count++$
- (10) foreach rule  $r$  in  $SFRc$
- (11)     if ( $r.count < minSupNum$ )  $SFRc = SFRc - \{r\}$
- (12) return  $SFRc$

**Fig. 1.** The local pruning algorithm

frequent rule by the *sentence-level constraint* without or only little loss of useful information for categorization.

Before presenting our methods, let us introduce the following definitions:

**Definition 1:** Given a document-level frequent rule  $r$  with an antecedent  $(i1, i2...im)$  and a document  $d$ , if  $(i1, i2...im)$  occurs in at least one same sentence of  $d$  (the order can be neglected), we say  $r$  satisfies *sentence-level constraint* and call it a *valid appearance* of  $r$ .

**Definition 2:** If the *valid appearances* of  $r$  in different documents of a separate training text collection meet a user-specified *sentence-level minimum support*, then the rule  $r$  is a *sentence-level frequent rule (SFR)*.

This pruning procedure is also performed category by category. *Valid appearances* of document-level frequent rules of category  $C$  are counted using training examples of their own category, and rules whose *valid appearances* are lower than the *sentence-level minimum support* are pruned. The details of this local pruning algorithm are described in Figure 1.

After the pruning phase, the number of rules is greatly reduced while the remaining rules become more meaningful from semantic perspective. This is clear because: firstly, the length of a document is often much larger than that of a sentence, and thus it's more likely for a rule to become document frequent than to become sentence frequent; secondly, the words included in the same sentence are semantically closer than those in same document.

### 3.2 Global Selection Phase

Though we introduce *sentence-level constraint* into pruning procedure in last section, we only evaluate rules by their *valid appearance* frequency. The accuracy of a rule has not been examined. To follow the convention in most association rule algorithms, we use the name *confidence* as a replacement of accuracy.

Given a set of training set  $D$  and a category  $C$ , let  $r$  be a *sentence-level frequent rule* of  $C$ , the *confidence* for  $r$  is defined as follows:

**Definition 3:** let  $D_r = \{d \mid r \subseteq s, s \subseteq d, d \in D\}$  and  $D_{r_c} = \{d \mid r \subseteq s, s \subseteq d, d \in D \text{ and } d \text{ is labeled as } C\}$ , then the confidence of  $r$ ,  $r.conf$ , is defined as:  $r.conf = |D_{r_c}| / |D_r|$

To calculate the *confidence*, we must merge all training examples from all categories together. In our implementation, we employ a global prefix tree to save *valid appearance* information of rules in different classes. Please note that the global tree is only used for efficient calculation in this step. Rules of each category are actually stored in a local prefix tree. Rules having a confidence lower than the sentence-level minimum confidence are removed since they often make wrong decisions.

The *confidence* measure represents the proportion of examples on which the rule is correct. It does not concern how the matching examples distribute in different classes. Entropy is a commonly used metric in information theory. It can measure homogeneity of examples the rule covers.

Suppose  $S$  is a collection of examples covering rule  $r$ . *Negative Entropy* is defined as follows:

**Definition 4:** *Negative Entropy*  $(S) = \sum_i^c P_i \log_2 P_i \cdot$  (1)

In (1),  $P_i$  is the proportion of  $S$  belonging to category  $i$ .

Since each  $P_i$  has been collected in the global prefix tree, the calculation of *negative entropy* has little extra overhead.

Algorithm: Global rule selection

Input: a set of sentence-level frequent rules of category  $c$ ,  $SFR_c$ ; a set of training documents,  $D$ ; a user-specified sentence-level minimum confidence,  $\text{minSentConf}$ ; a tuning factor,  $\sigma$ ; a coverage threshold,  $\lambda$

Output: the classifier of category  $c$ ,  $C_c$

Method:

- (1) Calculate confidence and negative entropy of each rule  $r$  in  $SFR_c$
- (2) foreach rule  $r$  in  $SFR_c$
- (3)     if( $r.\text{conf} \leq \text{minSentConf}$ )  $SFR_c = SFR_c - \{r\}$
- (4)  $C_c = \Phi$ ,  $\text{TrainSize} = |D|$
- (5) sort  $SFR_c$  according to their negative entropy in descending order
- (6) while( $SFR_c \neq \Phi$  and  $|D| \geq \lambda * \text{TrainSize}$ )
- (7)   set  $\text{topEntropy}$  and  $\text{bottEntropy}$  as the negative entropy of the first and last rule of  $SFR_c$
- (8)    $\text{tunevalue} = \sigma * (\text{topEntropy} - \text{bottEntropy})$
- (9)    $R = \Phi$
- (10)  foreach rule  $r$  in  $SFR_c$
- (11)     if( $r.\text{negentropy} \geq \text{topEntropy} - \text{tunevalue}$ )
- (12)          $R = R \cup \{r\}$
- (13)     else break
- (14)  set  $r_m$  as the rule having maximum coverage in  $R$
- (15)   $C_c = C_c \cup \{r_m\}$ ,  $SFR_c = SFR_c - \{r_m\}$ ,  $D = D - r_m.\text{coverdoc}$
- (16)  foreach rule  $r$  in  $SFR_c$
- (17)      $r.\text{coverdoc} = r.\text{coverdoc} - r_m.\text{coverdoc}$
- (18)     if( $r.\text{coverdoc} = \Phi$ )  $SFR_c = SFR_c - \{r\}$
- (19) return  $C_c$

**Fig. 2.** The global rule selection algorithm

We adopt sequential covering technology to select rules. The following principle governs the global selection phase: whenever there are choices, the rule having the highest negative entropy is of the first preference. Every time the “best” rule is selected, the examples covered by this rule are removed from current instances space. The process iterates until no more rules selected or a small number of examples uncovered.

A serious consideration is that when selecting the “best” rule, we check several rules with similar entropy simultaneously and chooses the rule having the highest coverage. Such a heuristic makes the search space shrink rapidly and avoids the overfitting problem caused by small disjunctions. We control the selection scope of rules with a parameter called *tuning factor*. On the other hand, since the rules with small database coverage are always noisy rules, we stop the covering procedure when the uncovered database space shrinks to a particular small range. A parameter called *coverage threshold* is introduced for this purpose. The details of this global selection procedure are shown in Figure 2.

In the algorithm, line 1 calculates the *confidence* and *negative entropy* of each rule  $r$  in original rule set. Line 2~3 remove the rules not satisfying the *minimum confidence threshold*. Line 4 initializes a container for selected rules. Line 5 sorts the rules by their *negative entropy* value. Line 6~18 describe the process of selecting rules. Line 7~13 chooses several rules which will compete for being selected. Line 14 finds the rule having the maximum coverage among the chosen rule set. Line 15 pushes the selected rule into the container and updates some variables. Line 16~18 update the covering document list of each rule and remove the rules which cover no document.

## 4 Predicting a New Document

The first issue that should be considered in prediction is whether to use multiple rules or not. We believe that a decision made by multiple rules is more reliable. So multiple rules work together to predict a test document in our method.

When predicting a new data, all current methods such as CBA, CMAR, ARC-BC did not involve how many times a rule matching the new data. We note that such an approach is no longer appropriate in text categorization since the multiple occurrences of words often imply that these words are actually the center of discussion. Just try to find the number of phrases “association rule” and “text classification” in this paper! So we think the number of a rule satisfying the test data should contribute to the classification procedure. Furthermore, because the rules have been filtered by a *sentence-level constraint* in our pruning phase, each sentence in test document should be treated as a matching unit.

To conform to the ideas introduced previously, we first split an unlabeled document into several sentences just as we do in the training phase. Each sentence is viewed as a transaction. Given a test document  $d=\{S_1, S_2, \dots, S_k\}$  and the classifier of category  $m$ ,  $C_m=\{R_1, R_2, \dots, R_n\}$ , for each sentence  $S_i$  of  $d$ , if there exists a rule  $R_j$  ( $R_j \in C_m$ ) which satisfies  $R_j \subseteq S_i$ , we say that  $S_i$  matches rule  $R_j$ . The matching score increases by the confidence of  $R_j$  when  $S_i$  matching rule  $R_j$ . If the total matching score of the category  $m$  is higher than a *classification minimum support*,  $d$  will be labeled with category  $m$ . Figure 3 illustrates the classification algorithm.

Algorithm: justifying whether a document should be labeled with a category

Input: a document,  $d$ ; a category,  $c$ , and its classifier  $C_c$ ; a user-specified classification minimum support,  $clsminsup$

Output: Yes/No

Method:

```

(1) supScore = 0, SenNum = |d|
(2) foreach transaction s in d
(3)   foreach rule r in  $C_c$ 
(4)     if( $r \subseteq s$ )
(5)       supScore = supScore + r.conf
(6) if(supScore/SenNum < clsminsup)
(7)   return No
(8) else
(9)   return Yes

```

**Fig. 3.** Labeling a new data algorithm

In the algorithm, line 1 initiates some variables, line 2~5 count the number of sentences which match the rules in  $C$  and increase the *supScore*. Line 6~9 make decision.

## 5 Experimental Results and Performance Study

Our experiment evaluations were done on the well-known Retuers-21578 dataset[12]. The ten most popular categories are selected for our evaluation. We also followed the *ModApte* split in which about 75% of the articles were used to build classifiers and the rest to test the accuracy of the classifiers. A stopword removal and term filtering procedure according to a given list of stopwords and the TF/IDF value of a term was done when generating document-level frequent rules.

The measures used in our experiment are *BEP*, *micro-BEP* and *macro-BEP*. The *BEP* (breakeven point) is the point at which precision equals recall. All these measures are obtained as reported in [5].

### 5.1 Parameter Settings

All the important parameters used in our classification method are summarized as follows:

- *minSentSup*: sentence-level minimum support (see section 3.2), which is used for mining *sentence-level frequent rule*. Typical values of *minSentSup* are around 10% ~ 15%.

- *minSentConf*: sentence-level minimum confidence (see section 3.3). Typical values of *minSentSup* are around 75% ~ 85%.
- $\sigma$ : tuning factor (see section 3.3). Typical values of tuning factor are around 20%~30%.
- $\lambda$ : coverage threshold (see section 3.3). Typical values of coverage threshold are around 5%~10%.
- *clsminsup*: classification minimum support (see section 4). Typical values of *clsminsup* are around 10%~15%.

Among these parameters, *minSentSup* seems to be the most crucial one and *clsminsup* the second and the others the third. This is because *minSentSup* limits the number of sentence-level frequent rules, which lay the foundation of the subsequent selection phase, and *clsminsup* directly decides whether a document belongs to a category or not.

### 5.2 Experimental Results

We first show the impacts of *minSentSup*. Figure 4 shows how the number of SFRs reduces with the increase of *minSentSup*. Number 3547 and number 50937 of DFRs in Fig.4 are gotten by setting document-level minimum supports 5% and 10% respectively.

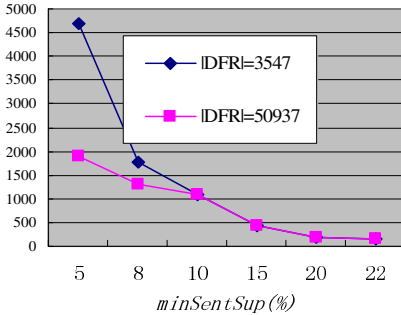


Fig. 4. Effect of *minSentSup* on |SFRs|

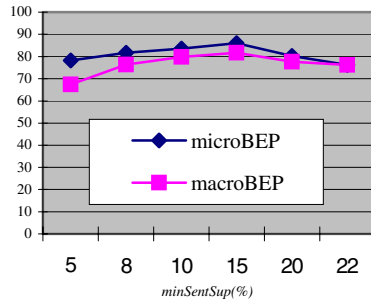


Fig. 5. Effect of *minSentSup* on BEPs

It is clear from Fig.4 that sentence-level constraint greatly prunes DFRs, and when *minSentSup* is set above a certain point (about 10%), the numbers of SFRs tend to be identical. Figure 5 shows how *minSentSup* impacts classification result. If we set *minSentSup* too low, we will find many useless rules; if we set it too high, we may miss important rules. Both these two settings have bad effects on the quality of the end classifier, as is illustrated in Fig 5. Other parameter settings in Fig.5 are: *minSentConf*=76%,  $\sigma$  =25%,  $\lambda$  =7%, *clsminsup* =13%.

In Table 1, we use the current best result of our 2-PS method to compare with other well-known methods. The results for the other classification systems are obtained from [5]. The parameter settings (*minSentSup*=14%, *minSentConf*=76%,  $\sigma$  =25%,  $\lambda$  =7%, *clsminsup*=13%) result in the best classification presented in Table 1. The comparison shows that our system outperforms all the other algorithms except SVM. While

compared to SVM and most of the other classifiers, our system can achieve much higher efficiency and higher scalability to deal with a large amount of documents. Our algorithm is a little slower than *ARC-BC* because we set a lower support threshold when mining DFRs (document-level minimum support 5% compared with 10% in *ARC-BC*), it also because our method need scan each sentence of each document while *ARC-BC* only need scan the bag of words.

In our experiments, our method is coded in a few thousand lines of C++, and it is run on a PentiumIV 1.7 GHz processor PC running Windows2000. Table 2 reports time and memory needed for training and testing all the ten categories. The time includes I/O time.

**Table 1.** BEPs on 10 largest Reuters categories

Category	2-PS	ARC-BC	Bayes	Rocchio	C4.5	k-NN	Bigrams	SVM (poly)	SVM (rbf)
acq	85.2	89.9	91.5	92.1	85.3	92.0	73.2	94.5	95.2
corn	89.9	82.3	47.3	62.2	87.7	77.9	60.1	85.4	85.2
crude	79.4	77.0	81.0	81.5	75.5	85.7	79.6	87.7	88.7
earn	96.4	89.2	95.9	96.1	96.1	97.3	83.7	98.3	98.4
grain	92.1	72.1	72.5	79.5	89.1	82.2	78.2	91.6	91.8
interest	62.6	70.1	58.0	72.5	49.1	74.0	69.6	70.0	75.4
money-fx	77.1	72.4	62.9	67.6	69.4	78.2	64.2	73.1	75.4
ship	79.9	73.2	78.7	83.1	80.9	79.2	69.2	85.1	86.6
trade	78.1	69.7	50.0	77.4	59.2	77.4	51.9	75.1	77.3
wheat	87.3	86.5	60.6	79.4	85.5	76.6	69.9	84.5	85.7
micro-avg	86.6	81.8	72.0	79.9	79.4	82.3	73.3	85.4	86.3
macro-avg	82.27	78.24	65.21	79.14	77.78	82.05	67.07	84.58	86.01

**Table 2.** Performance of 2-PS

<i>minSentSup</i>	Training		Testing	
	time	memory	time	memory
5%	38s	41.4M	11s	1.3M
8%	37s	41.2M	11s	1.3M
10%	36s	39.9M	11s	1.3M
15%	35s	39.0M	10s	1.3M
20%	35s	38.4M	10s	1.3M
22%	35s	38.2M	10s	1.3M

## 6 Conclusions

This paper proposes a novel 2-phase based method for associative text classification. The local pruning phase greatly reduces the number of association rules while keep the rules more semantically related and less redundant, and it also makes the following phase more efficient. The global selection phase selects the rules having largest differentiating ability by merging all examples from different categories. When classifying a new document, the times that a single rule matches the sentences in the



new document are under consideration in our method. The experimental results demonstrate that our approach can get better accuracy and performance than many other methods.

## References

1. Liu, B., Hsu, W., Ma, Y. Integrating Classification and Association Rule Mining. In Proc. 4<sup>th</sup> Int. Conf. Knowledge Discovery and Data Mining. New York (1998) 80-86
2. Li, W., Han, J., Pei, J. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In Proc. of the 1st IEEE International Conference on Data Mining .San Jose California (2001) 369-376
3. Bing Liu, Yiming Ma, Ching-Kian Wong, and Philip S. Yu. Scoring the Data Using Association Rules. Applied Intelligence, Vol. 18, No. 2, (2003) 119-135
4. D. Meretakis, D.Fragoudis, Hongjun Lu, S. Likothanassis. Scalable Association-based Text Classification. In Proc. of the 9<sup>th</sup> ACM International Conference on Information and Knowledge Management. McLean USA (2000) 5-11
5. Maria-Luiza Antonie, Osmar R. Zaiane. Text Document Categorization by Term Association. In Proc. of the IEEE International Conference on Data Mining. Maebashi City Japan (2002) 19-26.
6. Bing Liu, Wynne Hsu and Yiming Ma. Pruning and Summarizing the Discovered Associations. In Intl. Conf. on Knowledge Discovery and Data Mining. (1999) 125-134.
7. Davy Janssens, Geert Wets, Tom Brijs, Koen Vanhoof. Integrating Classification and Association Rules by proposing adaptations to the CBA Algorithm. Proceedings of the 10th International Conference on Recent Advances in Retailing and Services Science. Portland Oregon (2003)
8. Ke Wang,, Senqiang Zhou, Shiang Chen Liew. Building Hierarchical Classifiers Using Class Proximity. Proceedings of the 25th International Conference on Very Large Data Bases. (1999) 363 – 374
9. Mohammed J. Zaki, Charu C. Aggarwal. XRules: An Effective Structural Classifier for XML Data. The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington DC USA (2003)
10. J. Feng, H. Liu, and Y. Feng. Sentential Association Based Text Classification Systems. Proceeding of the 7th Asia Pacific Web Conference. Shanghai China (2005)
11. Ricardo Baeza-Yates, Berthier Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley (1999)
12. <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

# Intrusion Detection via Analysis and Modelling of User Commands

Matthew Gebski and Raymond K. Wong

National ICT Australia and School of Computer Science & Engineering,  
University of New South Wales, Sydney, NSW 2052, Australia

**Abstract.** Since computers have become a mainstay of everyday life, techniques and methods for detecting intrusions as well as protecting systems and data from unwanted parties have received significant attention recently. We focus on detecting improper use of computer systems through the analysis of user command data. Our approach looks at the structure of the commands used and generates a model which can be used to test new commands. This is accompanied by an analysis of the performance of the proposed approach. Although we focus on commands, the techniques presented in this paper can be extended to allow analysis of other data, such as system calls.

## 1 Introduction

There has been a flurry of activity recently in the area of intrusion detection. Viruses, hackers, malware and resource misappropriation are major issues that corporations of all sizes and end users face on a daily basis. Historically, in order to detect problems of this nature, system administrators and network analysts have been required to spend many hours poring over system logs and reports. In the last decade, there has been a considerable amount of attention placed on matching signatures of intrusion attempts. Following this, researchers have been examining more general ways to detect intrusions of a type not previously seen by the IDS.

In this paper, we concentrate on the problem of identifying inappropriate use of a user's account. The inappropriate usage may be a result of a number of causes included, but not limited to a malicious person:

- Using that user's terminal/workstation without permission
- Having gained access to a user's password
- Gaining access through security vulnerability and masquerading as a trusted user

Our aim is to mitigate risks that may be posed by such a malicious user. We are interested in analyzing the commands that user's execute to determine the likelihood of a given new command being issued by the 'true' user or by an intruder. This is done by examining both the content and ordering of the issued commands. In a similar fashion to other approaches, we attempt to create a

model of each user's actions, however, we use a tree based model over a clustering or IBL technique to better use sequence information.

The remainder of this paper is outlined as follows, we begin by providing an overview of the approaches that have been used for intrusion detection in general with particular emphasis on approaches that analyze user command data as well as a summary of our contributions. This is followed by our proposed approach. In Section 5, we examine our approach experimentally in addition to providing a complexity analysis and discussion. Finally, we conclude with a look at possible future work.

## 2 Related Work

Several influential approaches are relevant to our observation of intrusion detection systems. Many approaches emphasize detecting signatures indicative of impending or former intrusions, while more recent approaches are based on data mining techniques. We first consider some of the standard IDS techniques for dealing with network traffic. This is followed by an overview of user log based approaches.

**General IDS.** Existing IDS have been aimed at detecting aberrant network traffic with a combination of 'raw' metrics such as amount of data transferred and 'application' metrics such as number of failed login attempts. Typically, these systems have been based on pattern matching, [6], [1], [2]. Lately, other approaches have arisen such as using artificial neural networks (ANNs) [11]. Other approaches include using association rule techniques, [9], episode rules [5] and root cause analysis [4].

**User Log IDS.** In comparison to dealing with network data, detecting intrusions via user logs is particularly interesting due to the difficulty in obtaining real world examples of attacks. Moreover, it is difficult to create signature based approaches due to the practically limitless number of command combinations that may arise. There have been two main projects that have examined user based intrusion detection. Firstly, Lane and Brodley's instance based learning (IBL) approach, [8] [7], creates profiles for users based on the similarity between the user's history. Due to the number of possible commands that could be entered and the lack of labeled data, other IBL techniques such as nearest neighbor which have previously been shown to perform well (some of the best performing approaches in the intrusion detection KDD Cup from 1998 were based on the 1 nearest neighbor approach) were unsuitable for this task. The similarity between two sequences is defined as the number of slots that match between two sequences. Comparisons are then made between the previous sequence structure from the user and a sequence to be tested. If the difference between the two is greater than a pre-determined threshold, the test sequence is classified as a potential intrusion.

The ADMIT system by Sequeira and Zaki [12] manages a profile for each user of the system, with each profile being comprised of a set of clusters that are

designed to characterize a user's habits. The distance between two sequences of commands is based on a sub-sequence matching function such as Longest Common Subsequence (LCS) or Match Count Polynomial Bound (counts the number of slots in which the two sequences are identical). Clusters are then refined based on a threshold value. Unfortunately, there is no clear way to determine this value with experimentation being the suggested method.

Tests for intrusion are performed by locating the closest cluster for a given test sequence and a comparison is made between previous sequences and the test sequence. If the rating for the audit sequence is below a threshold (separate from the cluster refinement threshold), the test sequence is marked as possible intrusion.

Other application level approaches include monitoring various system metrics [13] and re-authentication techniques [10]. In addition to intrusion detection from user commands, there has also been some preliminary work into systems for generating attacks given that the algorithm for the IDS is known - intuitively, if an intruder mimics typical user behavior, it will be difficult to detect [15].

### 3 Data Model

For the remainder of the paper, we assume the commands being considered have been entered into a Unix terminal. However, there is no reason that our model can not be generalized to other systems. A user has multiple command sessions, each containing a number of commands entered into a computer system. We model the user as a list of sessions, each session containing a list of commands. Currently, we only scrutinize the commands that are directly entered and do not consider commands that may be run as a secondary component of another process. If a program  $\rho$  requires a directory listing as part of its execution, reads from a file in that directory and finally writes to the file, we only consider the initial execution of  $\rho$ . As part of our preprocessing measures, we clean the logs, this ensures consistency with other approaches and helps ensure the privacy of users. The first file name to be mentioned in a session is replaced by  $\langle 1 \rangle$ , the second with  $\langle 2 \rangle$  and so on. This means that there is no way to compare the usage of a file between sessions, file  $\langle 1 \rangle$  in one session may be  $\langle 6 \rangle$  in the subsequent session.

The database of logs being considered,  $D$ , is composed of a number of users,  $U_0, U_1 \dots U_n$ . Each user is composed of a sequence of sessions  $U_{S_1}, U_{S_2}, U_{S_3} \dots U_{S_j}$ . In turn, each session,  $S_j$  contains a sequence of commands  $S_{jC_k}$ . Unless otherwise noted, we will only be considering one session at a time. Additionally, when appropriate, we will simply refer to  $C_1, C_2 \dots C_k$  when there is no ambiguity as to which session the commands belong to. We will use  $C_k < C_{k+1}$  to represent  $C_k$  occurring before  $C_{k+1}$ . We define the distance between two commands,  $C_x$  and  $C_y$  as  $\delta(C_x, C_y) = 1 + |\{C_j | C_x < C_j < C_y\}|$  - conceptually, this is the number of commands that have between entered between  $C_x$  and  $C_y$ . Finally, for two commands  $S_{iC_j}$  and  $S_{xC_y}$  from sequences  $S_i$  and  $S_x$  are equal if their values are the same and is denoted as  $S_{iC_j} = S_{xC_y}$ . An example of this would be if both had the value  $g++$ .

After processing, we build a profile  $\rho_U$  that allows us to test if a specific sequence of test commands,  $Q$ , have been created by the same user modelled by  $\rho_U$ , or some (presumably hostile) third party.

## 4 Our Approach

We have the following desiderata: First and foremost, we would like to be able to compare two different sessions,  $S_X$  and  $S_Y$  against the profile model for a user,  $\rho_{U_i}$ , and determine which of these is most likely to have been drawn from the same distribution as the one used to create  $\rho_{U_i}$ . Secondly, our approach must be robust in terms of its ability to handle unseen commands - in general, we do not wish for the system to raise an alarm every time a user decides to run a program that they have never previously run. Finally, it would be ideal for our system to handle online queries, and as such we wish for the scalability and faithfulness of the model to facilitate such operations.

Let us consider the following hypothetical sequence of events recorded for hypothetical user  $U$ :

1.  $C_1$  -  $U$  examines the contents of the current directory : *ls -la .*
2.  $C_2$  -  $U$  opens a file  $f$  : *vi f*
3.  $C_3$  -  $U$  then compiles  $f$ : *g++ f*

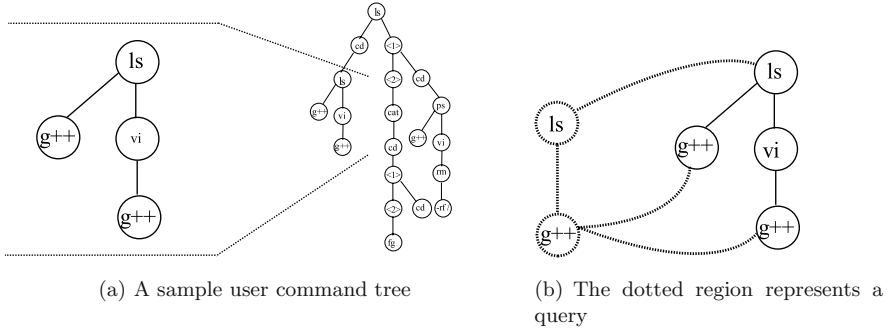
Intuitively, we would be surprised if any of these actions were independent of each other, in fact for the remainder of this paper, our work will be based on the assumption that they are dependent.

This motivates us to look for a way to model how each command is affected by prior commands. Returning to our example, we wish to typify that  $C_1$  affects both  $C_2$  and  $C_3$  and that  $C_2$  affects  $C_3$ . We choose to represent the commands as a tree; representing the commands as a tree allows us to say that commands further down the tree are dependent on those further up the tree (with limitations which we will discuss later). Each node of the tree represents a command, while the weights of the edges between nodes represent probabilities between these commands. Figure 1 shows the tree, without weights (for clarity), for the aforementioned example, *ls -la .; vi f; g++ f*. The input to the tree building algorithm is a user  $U$  and the output is a profile  $\rho_U$  in the form of a tree  $T_u$ .

We can choose the suffix tree data structure as to represent the commands. Instead of treating each word of a document as a collection of tokens (characters), we treat each session as a collection of command tokens. This allows our tree to be constructed in  $O(n)$  time using Ukkonen's construction algorithm [14].

### 4.1 Scoring Unseen Command Sequences from Profiles

Given that we have a tree,  $T$ , representing the history of commands, we now consider how a unseen command sequence,  $Q$ , could be matched against  $T$  in order to determine if the same user was responsible for the commands that lead to both  $T$  and  $Q$ . Unlike the construction procedure for  $T$ , we do not build a



**Fig. 1.** A sample user command tree and query being executed

tree for  $Q$  and  $Q$  is simply the list of commands. The following cases may arise when matching  $Q$  to  $T$ :

1.  $Q$  exactly matches a path from the root to a leaf in  $T$ . This will happen when we have seen the sequence of commands in  $Q$  as a sequence or subsequence at one stage in one of the sessions processed when constructing  $T$ . This is the most straightforward case to handle as we can directly compute the score of  $Q$  from the path in  $T$ .
2. Given a command  $Q_{C_i}$  and  $Q_{C_{i+1}}$  that directly follows  $Q_{C_i}$  in  $Q$ , there is no corresponding sequence in  $T$ , but there a  $T_x$  such that  $T_x = Q_{C_i}$  and a  $T_y$  such that  $T_y = Q_{C_{i+1}}$  and  $T_x < T_y$ . An example of this would be  $T$  being formed by  $vi; ls; g++$  and  $Q$  being  $vi; g++$ .
3. The opposite of the previous case in which we have a  $Q_{C_i}$  and  $Q_{C_j}$  such that  $C_i < C_j$  and  $Q_{C_i} = T_{C_x}$  and  $Q_{C_j} = T_{C_{x+1}}$ .
4. None of the entire sequence  $Q_0$  to  $Q_n$  matching  $T$ . While in many real world scenarios, we would not expect this case to occur frequently, we should not be surprised to see this case arise when a user begins a new project or starts using a new program.

Given a query  $Q$  and profile  $\rho$  based on the command tree  $T$ , we first calculate the paths in  $T$  that are *relevant* to  $Q$ . A path,  $T_p$ , is considered relevant if any of the first three cases are satisfied. The query is then divided into pairs, which are matched against the tree. The score for an individual path in  $T_p$  against a pair  $Q_x, Q_y$  from  $Q$  is:

$$Score(T_p, Q_x, Q_y) = \lambda * |\delta(T_p[1], T_p[last]) - \delta(Q_x, Q_y)|$$

The total score for the unseen sequence  $Q$  with the set of paths  $P = \{T_p | T_p \text{ is relevant to } Q\}$  is calculated as:

$$R_Q = \frac{1}{\binom{n}{2}} \sum_{x=1}^n \sum_{y=x+1}^n \sum_{i=1}^{|P|} \frac{Score(P[i], Q_x, Q_y) * \alpha_{P[i]}}{|P|}$$

**Algorithm 1** Matches query  $Q$  against history tree  $T$ 


---

```

MATCH SEQUENCE ( $T, Q$ )
1:  $pairs = gen\_pairs(Q)$ 
2:  $scores = List()$ 
3:  $tc = 0$ 
4: for  $(Q_x, Q_y) \in pairs$  do
5:   if  $T.contains\_path(Q_x, Q_y)$  then
6:      $P = T.get\_paths(Q_x, Q_y)$ 
7:      $res = CalcPathScore(Q_x, Q_y, P)$ 
8:      $tc = tc + res.pc$ 
9:      $scores.append(res)$ 
10:  else
11:     $scores.append(T_{new}, 1)$ 
12:     $tc = tc + 1$ 
13: return  $AggregateScores(scores, tc)$ 

```

---

**Algorithm 2** Calculates the score and count for a path

---

```

CALCPATHSCORE ( $Q_x, Q_y, P$ )  $pc = 0$ 
1: for  $p \in P$  do
2:    $val = Score(p, Q_x, Q_y)$ 
3:    $pc = pc + p.count$ 
4: return  $(val, pc)$ 

```

---

$\lambda$  is a damping function that is used to control the rate at which older commands influence the probability of newer commands. There are a number of functions that are potentially suitable as damping functions for the weights of the edges.  $\frac{1}{x^2}$  in many cases is too harsh, with command  $C_i$  being effectively unaffected by  $C_j$  after approximately  $j - i > 3$  which often leads to inconsistencies with our model. Conversely,  $\frac{1}{x}$  provides a reasonable drop off, allowing commands as late as  $C_{i+10}$  to contribute, but 'encouraging' the earlier commands  $C_{i+1}$ ,  $C_{i+2}$ ,  $C_{i+3}$ , to dominate.

In situations where no paths are considered relevant, we use a default constant  $T_{new}$  which is the probability of seeing a command that has not been previously seen - in practice, we simply divide the number of different command tokens that exist by the total number of commands seen. It is essential to have this value to ensure that new commands are not automatically flagged as intrusion attempts.

When calculating the final score during the sequence matching process, we moderate each of the weights by the number of times that the path has observed as a percentage of the total number of paths that are relevant to the given query. That is, for the set of matching paths  $S$ , and a given set of paths  $p$  we refer to  $\alpha_p = \frac{p.count}{\sum \{p.count: p \in S\}}$  as the *contribution* of  $p$ .

Given a score for a query, we are then able to decide if the query is likely to be an intrusion. Similar to both Sequeira and Zaki [12] and Lane and Brodley [8], we use a threshold value to decide if a query should be flagged. The threshold

---

**Algorithm 3** Aggregates the calculated scores

---

```

AGGREGATESCORES (scores, totalcount)
1: sum = 0
2: for s ∈ scores do
3:   sum = sum + (s.val * s.pc)
4: res = sum/totalcount
5: return res

```

---

can be initially determined by examining a number of valid sessions and setting the threshold such that these sessions would not trigger an alarm. This can be performed using the same training data used for construction of the tree, or alternately, withholding a portion of the training data solely for threshold determination.

## 4.2 Allowing Decay

Let us consider a user writing a research paper on IDS. Many of the commands during a session may be related to development such as *vi*; *g++*; *gdb*, or to writing the paper, *vi*, *latex*, *bibtex*. Following this period, the user may begin using other commands unrelated to either development or paper writing. Accounting for such concept drift is important for the IDS to perform correctly. Given the current 'time'  $t$  and an age of a path  $p$ , we define the age of  $p$ ,  $Age(p) = (p.t - t)$ . We redefine the contribution of a set of commands as:  $\alpha'_p = \alpha_p * Age(p)^{-d}$  Where  $d$  is the *age damping* function. Alternately, we could use the decay system proposed by Sequeira and Zaki:  $\beta_j = \frac{\beta_{j-1}}{\beta_{j-1} + 1 - \log(\frac{z}{y+j})}$ ,  $1 - \log(\frac{z}{y+j}) > 0$

## 5 Experimental Evaluation and Analysis

**Experimental Set Up.** For our experiments, we use the Purdue Unix User Data (available from [3]) set previously used by Lange and Brodley as well as Sequeira and Zaki. This contains nine usage session histories from eight users with one user providing two sets (this user was using two different platforms for separate work on two projects). We set out to measure the ability of our approach to take sessions and classify them as intrusions or normal cases.

For testing intrusions, we draw sessions from all users and compare them against all users except the one from which the test sequence was selected. That is, we test a command sequence from USER1 against all users except USER1. Additionally, we test the ability of a user's classifier to determine if sessions from that user are intrusions.

**Accuracy Results.** We begin by looking at an example in which we have constructed each user's profile from one hundred sessions from that user. Each user is then tested against the hundred sessions from USER6. The results of



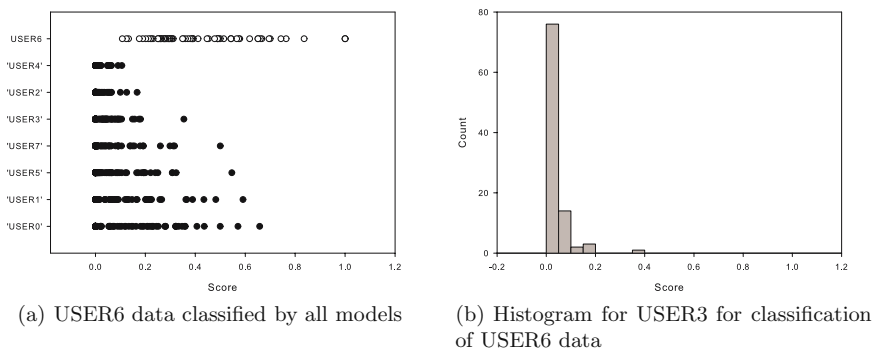


Fig. 2. Classification against USER6 sessions

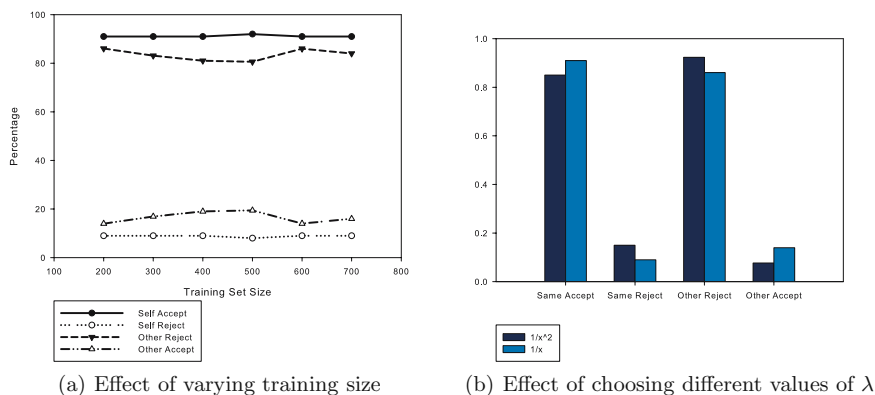


Fig. 3. Effect of modifying training size and  $\lambda$

this test can be seen in Figure 2a - black dots indicate the score a user received against a test from USER6, while white dots indicate the score that USER6 received for the tests against itself. We can see that there is a clear difference in the scores for USER6 compared to the other seven users. The histogram in Figure 2b, shows distribution of USER3, as we can see for a large number of cases (approximately 80%), USER3 scored very close to 0 and almost all cases had a score lower than 0.1.

**Effect of Varying Training Size.** The amount of audit data used for training is an important factor in assessing the efficacy of our approach. Models were constructed from samples of one hundred sessions and upwards. No decay function was used for these experiments. Self Accept refers to the model accepting a session that was used as part of the construction process, Self Reject refers to the model accepting a session that it shouldn't. Conversely, Other Reject refers to a model correctly classifying an intrusion, while Other Accept refers to a model

incorrectly classifying an intrusion as benign. Figure 3a, shows that as training size increases, there is a minimal change in quality in terms of Self results, overall we receive under 10% false positives. However, it is interesting to note that the performance in terms of intrusions classifications begins to decline as the number of sessions increases. This is a result of the increase in the number of tokens that occur - we can alleviate this problem by being decaying older sessions. We still classify slightly 85-88% of sessions correctly.

**Effect of Damping Function.** Experiments were performed with various damping functions,  $1/x$  and  $1/x^2$ . As seen in Figure 3b,  $1/x$  provides much better performance for Self Accepts and Self Rejects, while  $1/x^2$  increases the quality of the Other test cases.

**Analysis of Misclassified Points.** It is important for us to examine which types of queries are typically misclassified by our technique. In terms of cases being wrongfully labelled as not being intrusions, this occurs from similar usage habits. For example, the right most point in Figure 2 for USER7 represents a sequence similar to a sequence that we would expect from USER6.

## 6 Discussion

Compared to Sequeira and Zaki who reported approximately 80% intrusion detection and 15% false positives, and Lane and Brodley who reported approximately 74% and 28% respectively, our results are very promising. Our approach provided better coverage for existing intrusions, approximately 85% while at the same time, providing under 10% false positives. For the most part, the intrusion queries we missed were due to sessions that were due to sessions using only very common tokens such as *cd*, *ls*, etc.

Naturally, we would expect a tool of this sort to be implemented at the kernel level in practice. This avoids problems with users using other programs or scripts with malign intentions. Additionally, this provides a mechanism to lock out or provide a secondary challenge, such as another password, to an intruder in real time allowing mitigation to be performed preemptively before any serious damage actually occurs.

## 7 Conclusions and Future Work

In this paper, we have presented a new approach for construction of a tree based model for intrusion detection analysis of user command histories. This model is intuitive and based on the probabilities of a command being affected by prior commands. The tree based model presented here makes use of the sequential structure of the data to better facilitate analysis. Our preliminary experimental evaluation has demonstrated that our technique is competitive with cluster based approaches, but without the need for the number of clusters to be known *a priori*.

Our future work concentrates on better understanding the relationships between the relative age of commands and their relevance to each other. We hope

this will allow us to construct more accurate models that can more faithfully represent the implicit structure of command histories. Additionally, we aim to make our approach more robust in terms of masqueraders with access to earlier portions of the command logs to reduce masking of inappropriate use.

## References

1. S. Antonatos, K. Anagnostakis, M. Polychronakis, and E. Markatos. Performance analysis of content matching intrusion detection systems. In *Proceedings of the 4th IEEE/IPSJ Symposium on Applications and the Internet*, pages 25–30, 2004.
2. C. R. Clark and D. E. Schimmel. A pattern-matching co-processor for network intrusion detection systems. In *IEEE International Conference on Field-Programmable Technology (FPT)*, pages 68–74, Tokyo, Japan, 2003.
3. S. Hettich and S. D. Bay. The UCI KDD archive - <http://kdd.ics.uci.edu>, 1999.
4. K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Trans. Inf. Syst. Secur.*, 6(4):443–471, 2003.
5. K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 366–375, New York, NY, USA, 2002. ACM Press.
6. S. Kumar and E. H. Spafford. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference*, pages 11–21, 1994.
7. T. Lane and C. E. Brodley. Approaches to online learning and concept drift for user identification in computer security. In *Knowledge Discovery and Data Mining*, pages 259–263, 1998.
8. T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Trans. Inf. Syst. Secur.*, 2(3):295–331, 1999.
9. W. Lee. Applying data mining to intrusion detection: the quest for automation, efficiency, and credibility. *SIGKDD Explor. Newsl.*, 4(2):35–42, 2002.
10. M. Pusara and C. E. Brodley. User re-authentication via mouse movements. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 1–8, New York, NY, USA, 2004. ACM Press.
11. J. Ryan, M.-J. Lin, and R. Miikkulainen. Intrusion detection with neural networks. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.
12. K. Sequeira and M. Zaki. Admit: anomaly-based data mining for intrusions. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 386–395. ACM Press, 2002.
13. J. Shavlik and M. Shavlik. Selection, combination, and evaluation of effective software sensors for detecting abnormal computer usage. In *KDD '04: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 276–285, New York, NY, USA, 2004. ACM Press.
14. E. Ukkonen. Constructing suffix trees on-line in linear time. In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1*, pages 484–492. North-Holland, 1992.
15. D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 255–264, New York, NY, USA, 2002. ACM Press.

# Dynamic Schema Navigation Using Formal Concept Analysis

Jon Ducrou<sup>1</sup>, Bastian Wormuth<sup>2</sup>, and Peter Eklund<sup>1</sup>

<sup>1</sup> School of Economics and Information Systems,  
The University of Wollongong,

Northfields Avenue, Wollongong, NSW 2522, Australia  
jrd990@uow.edu.au, peklund@uow.edu.au

<sup>2</sup> Darmstadt University of Technology, Department of Mathematics,  
Schloßgartenstr. 7, 64289 Darmstadt, Germany  
bastian@wormuth.info

**Abstract.** This paper introduces a framework for relational schema navigation via a Web-based browser application that uses Formal Concept Analysis as the metaphor for analysis and interaction. Formal Concept Analysis is a rich framework for data analysis based on applied lattice and order theory. The application we develop, D-SIFT, is intended to provide users untrained in Formal Concept Analysis with practical and intuitive access to the core functionality of Formal Concept Analysis for the purpose of exploring relational database schema. D-SIFT is a Web-based information systems architecture that supports natural search processes over a preexisting database schema and its content.

## 1 Introduction

This paper presents a new application framework for relational schema navigation using Formal Concept Analysis (FCA). The idea behind the framework is the simplification of existing application development frameworks for FCA, in particular the way humans process standard searches in FCA. The software prototype – called D-SIFT (Dynamic Simple Intuitive FCA Tool) – consolidates various features that have been introduced by other applications [6,1] but is a more general framework. D-SIFT allows users to define query elements based on database schema and contents, using one of two query modalities, to visualise structures in the data as a concept lattice. The interface allows dynamic creation of lattice diagrams and allows the user to add and remove attributes from the displayed concept lattice according to preference. D-SIFT implements the classical features of FCA software with so-called *mandatory attributes*. The user is able to restrict the displayed object set by the selection of mandatory attributes. The resulting concept lattice is limited to objects which share these attributes. This process closely resembles iterative search in information retrieval, where the user starts from one or two keywords and progressively refines the result set by the addition of further (or different) keywords. Further, D-SIFT is more easily accessible as a

platform than existing FCA frameworks. The required plug-ins used are provided in standard configurations of most Web browsers, and the underlying database complies with the CSV file format (text files with comma-separated entries).

## 2 Formal Concept Analysis Background

Formal Concept Analysis [5] has a long history as a technique of data analysis that when applied conforms to the idea of Conceptual Knowledge Processing. Data is organized as a table and is modeled mathematically as a many-valued context,  $(G, M, W, I_w)$  where  $G$  is a set of objects,  $M$  is a set of attributes,  $W$  is a set of attribute values and  $I_w$  is a relation between  $G$ ,  $M$ , and  $W$  such that if  $(g, m, w_1) \in I_w$  and  $(g, m, w_2) \in I_w$  then  $w_1 = w_2$ . In the table there is one row for each object, one column for each attribute, and each cell is either empty or asserts an attribute value.

Organization over the data is achieved via conceptual scales. A conceptual scale maps attribute values to new attributes and is represented by a mathematical entity called a formal context. A formal context is a triple  $(G, M, I)$  where  $G$  is a set of objects,  $M$  is a set of attributes, and  $I$  is a binary relation between the objects and the attributes, i.e.  $I \subseteq G \times M$ . A conceptual scale is defined for a particular attribute of the many-valued context: if  $\mathbb{S}_m = (G_m, M_m, I_m)$  is a conceptual scale of  $m \in M$  then we define  $W_m = \{w \in W | \exists (g, m, w) \in I_w\}$  and require that  $W_m \subseteq G_m$ . The conceptual scale can be used to produce a summary of data in the many-valued context as a derived context. The context derived by  $\mathbb{S}_m = (G_m, M_m, I_m)$  w.r.t. to plain scaling from data stored in the many-valued context  $(G, M, W, I_w)$  is the context  $(G, M_m, J_m)$  where for  $g \in G$  and  $n \in M_m$

$$gJ_m n \Leftrightarrow \exists w \in W : (g, m, w) \in I_w \text{ and } (w, n) \in I_m$$

Scales for two or more attributes can be combined in a derived context. Consider a set of scales,  $S_m$ , where each  $m \in M$  gives rise to a different scale. The new attributes supplied by each scale can be combined:

$$N := \bigcup_{m \in M} M_m \times \{m\}$$

Then the formal context derived from combining these scales is:

$$gJ(m, n) \Leftrightarrow \exists w \in W : (g, m, w) \in I_w \text{ and } (w, n) \in I_m$$

Several general purpose scales exist such as ordinal and nominal scales. A nominal scale defines one formal attribute for each value that a many valued attribute can take. An ordinal scale can be used on a many-valued attribute for which there is a natural ordering, for example,  $\text{size} \leq 20$ ,  $\text{size} \leq 40$  and so on. The derived context is then displayed to the user as a lattice of concepts. A concept of a formal context  $(G, M, I)$  is a pair  $(A, B)$  where  $A \subseteq G$ ,  $B \subseteq$

$M$ ,  $A = \{g \in G \mid \forall m \in B : (g, m) \in I\}$  and  $B = \{m \in M \mid \forall g \in A : (g, m) \in I\}$ . For a concept  $(A, B)$ ,  $A$  is called the extent and is the set of all objects that have all of the attributes in  $B$ , similarly,  $B$  is called the intent and is the set of all attributes possessed in common by all the objects in  $A$ . As the number of attributes in  $B$  increases, the concept becomes more specific, i.e. a specialization ordering is defined over the concepts of a formal context by:  $(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow B_2 \subseteq B_1$ . In this representation more specific concepts have larger intents and are considered “less than” ( $<$ ) concepts with smaller intents. The analog is achieved by considering extents, in which case, more specific concepts have smaller extents. The partial ordering over concepts is always a complete lattice [5]. For a given concept  $C = (A, B)$  and its set of lower covers  $(A_1, B_1) \dots (A_n, B_n)$  with respect to the above  $<$  ordering the object contingent of  $C$  is defined as  $A - \bigcup_{i=1}^n A_i$ . We shall refer to the object contingent simply as the contingent in this paper.

### 3 Conceptual Information Systems from Databases

D-SIFT takes a user-supplied *comma separated values* database (CSV) and provides an interface to the database as a conceptual information system. For this reason the input format of D-SIFT closely aligns with a typical export format from a relational database management system (RDBMS) and common applications like Excel and OpenOffice.

The CSV format is simple, easy to read and edit. It is a common optional output format for most modern and legacy applications and database systems. CSV files are forced to contain only data that can be expressed as text; this caters to the input requirements of D-SIFT. To translate the CSV database into a Conceptual Information System, the user indicates how D-SIFT should treat each field. This requires the user to indicate a field which is each entry’s identifier (entity in RDBMS terms) and then group the remaining fields into nominal or numerical scale models.

In order to extract objects with meaningful names, the user identifies the field which provides an identifier for the database (e.g. a candidate key such as *name* in a database of people). Nominal data, in FCA terms, is usually text (e.g. *names* or *locations* in the people database), and sometimes represents boolean values (e.g. attributes such as *gender* or attributes with values such as *yes/no*). Numerical data is represented by numbers which, over the scope of the entire field, have some form of ordering (e.g. a schema attribute such *length in metres* with some entries longer than others).

There are instances where database attributes with numeric data should not be scaled ordinally; for example identifiers such as social security numbers, which may or may not be indicative of an order. D-SIFT also gives the option to drop fields that are not of interest to the user, by tagging those fields (e.g. *comment* or *ID* fields). Interaction with the CSV file described to this point in the text allows D-SIFT to collect enough information to construct the context and scale information for the Conceptual Information System.

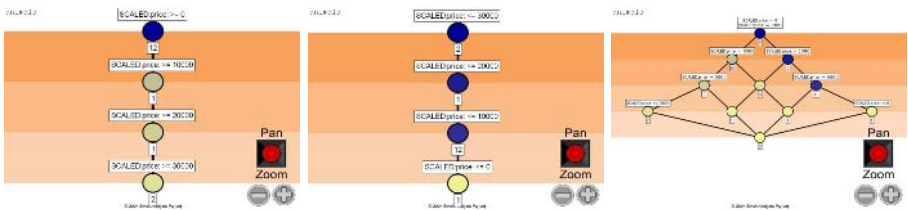
### 4 Using D-SIFT

D-SIFT intends to offer the user a flexible tool for viewing the various structures and relationships that are present in a database. The user only needs some understanding of the data they are viewing; enough to understand the objects being dealt with and the meaning of attributes, and some level of ability reading lattice diagrams. The owner of a database should know its content and user testing has shown that users can quickly become competent at reading lattice diagrams with little or no formal training [4].

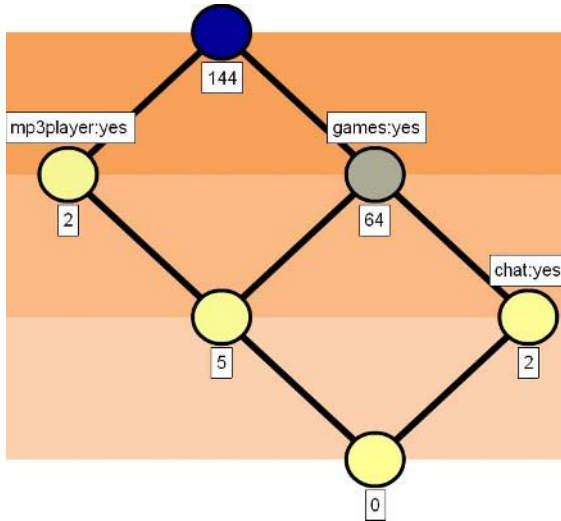
The user constructs queries by selecting query elements of interest and assigning them to one of two lists; *Zoom* or *Filter*. Query elements are made up of one or more *nominally-scaled* or *numerically-scaled* attributes. Nominally-scaled attributes comprise attribute groups and an attribute value. Numerically-scaled attributes comprise an attribute group, a *size* and an *order*. The size of a numerically scaled attribute can be thought of as the number of intervals which will be produced, while the order specifies the way in which the values should be compared. The orders are of three types, *Ordinal Up*, *Ordinal Down* and *Interordinal*. Ordinal Up and Down correspond to comparisons based on  $\geq$  and  $\leq$  respectively. Interordinal generates both Ordinal Up and Down (see Fig. 1).

The *Zoom* list should be populated with query elements that are ‘required’. The elements of the *Zoom* list are used to restrict the object set of the context to only objects with elements in the list. This is a conjunctive query so if dichotomous elements are in the *Zoom* list, the object set will be empty. Query elements in the *Zoom* list will always appear as attributes of the topmost concept of the diagram. Ordinal element groups cannot be added, nor can two element values from the same element group. The *Filter* list should be populated with query elements that are ‘of interest’. The elements in the *Filter* list are used to restrict the attribute set of the context. This means that only elements in the *Filter* list (and any from the *Zoom* list) will feature in the resulting lattice. These attributes are used to show structural relations in the database.

Using this query building paradigm of ‘required’ and ‘of interest’, the user can perform exploratory tasks against the database. The simplest example of which is the idea of a ‘search’ for an object that meets several criteria, or aids in the discovery of the ‘next best’ when the exact result is unavailable. In our



**Fig. 1.** Concept lattices showing the different types of numeric scaling available via the D-SIFT interface are (from left to right) Ordinal Up, Ordinal Down and Interordinal. All are shown with a size of 3.



**Fig. 2.** Diagram generated from the Phones database with mp3player:yes, games:yes and chat:yes as *Filter* elements.

examples, a database concerning cellphones, we imagine a potential customer of a new phone. The user may have a rough idea of the technical features but no understanding which of these he really needs or wants. The case scenario follows the user’s looking at all the features – or specifying known features. After obtaining an overview of the data, the user can sort the features into those that are essential and the remaining features as softer constraints on the search. The user may have already encountered dichotomous features, but not knowing which to eliminate may continue to use both. Step-by-step the user will make decisions and compromises before selecting the phone with the features that satisfy the search criteria. The last part of the search process will require many comparisons and iterations when exploring the information landscape with multiple dichotomous attributes.

As more *Filter* elements are added the complexity of the resulting lattice will most likely increase exponentially. To counter this complexity increase, which can make the diagram difficult to understand, elements of *Filter* can be promoted to *Zoom*. This will decrease the object set and decrease the number of attributes used to show structure of the data, which in turn reduces the complexity of the lattice diagram. The advantage of having the structure as a lattice is that the user can visualize relationships. Of these relationships it is easiest to see relations such as mutual exclusivity and implication. Figure 2 shows a simple lattice diagram. The user can see that mp3player:yes and chat:yes are mutually exclusive (there are no phones with both an MP3 player and a chat function) because the point where the concepts join (reading the diagram downwards) has an extent size of 0. Also, it can be seen that chat:yes implies games:yes (every phone with a chat function also has games). In a search context, where the desired result has all query elements in the *Filter* list, it can be seen that there are 0



total matches (bottom-most concept extent size is 0), but there are 7 phones that meet 2 of the query elements (the concepts directly above the bottom-most concept).

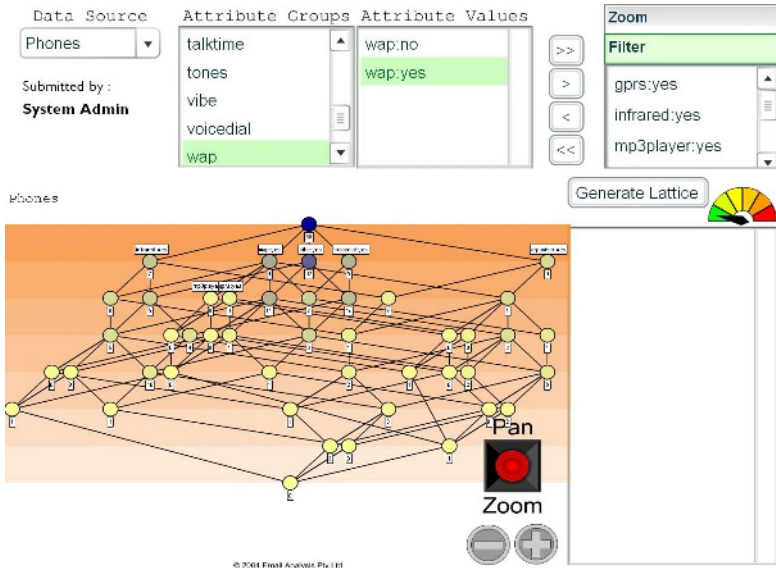
### 4.1 Case Scenario One: The Exploration Method

We now demonstrate the ideas described in the previous section with respect to a more concrete interaction scenario. In this scenario, the user knows every feature considered important (and desirable) in a new cellphone. In this case scenario the user wants:

*Infrared capabilities Built-in MP3 player Built-in organiser Vibration alert  
Voice-dial WAP support GPRS support*

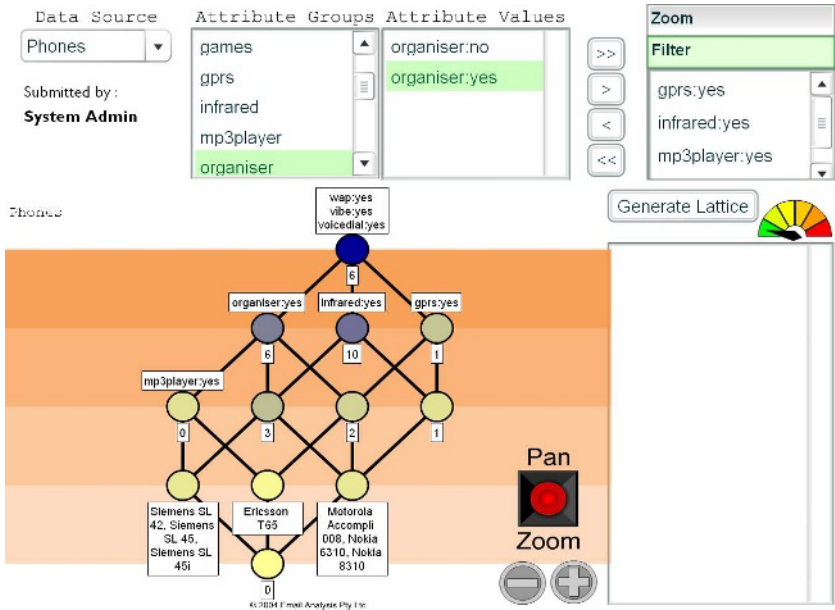
The user adds all the corresponding attributes as ‘filter’ attributes. The resulting line diagram from these filter attributes is too large for the user to make an instant decision, but the line diagram gives an overview of the search space and it is possible to conclude the following from it:

1. The top-most concept has a contingent of 60, therefore there are 60 phones with none of the desired features.
2. The bottom-most concept has an empty extent, therefore there is no phone with all desired features.
3. The attributes `vibe:yes`, `voicedial:yes` and `wap:yes` are most common in this diagram<sup>1</sup>.



<sup>1</sup> Recognizable by the fact they are darker in colour - indicating a large extent compared to other concepts. This is the coloring style used in TOSCANAJ [2].

This knowledge leads the user to zoom on the three common attributes, which would seem a good way to reduce the complexity of the data while still maintaining the majority of the phones.



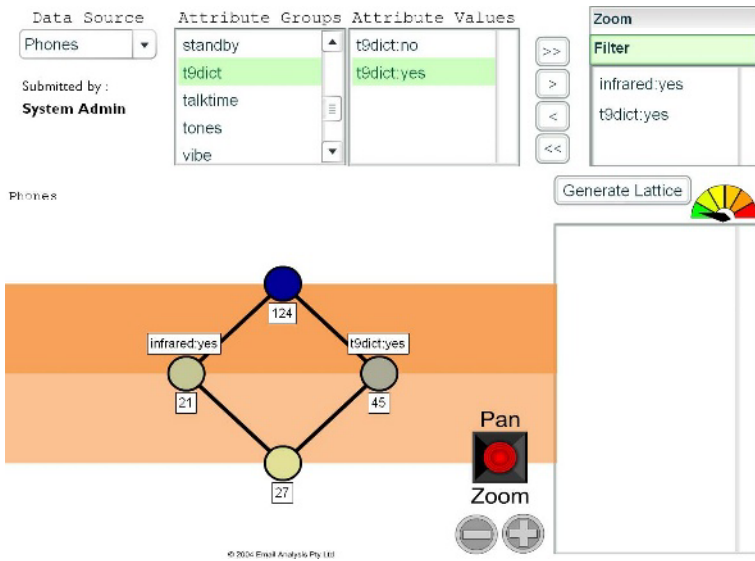
The result above shows that at least one desired feature can not be kept, and that the selection is from 7 phones in 3 groups – each group has one of the desired features missing.

- The *Siemens SL 42*, *Siemens SL 45* and *Siemens SL 45i* do not have GPRS Support.
- The *Ericsson T65* does not have infrared capabilities.
- The *Motorola Accompli 008*, *Nokia 6310* and *Nokia 8310* do not have a built-in MP3 player.

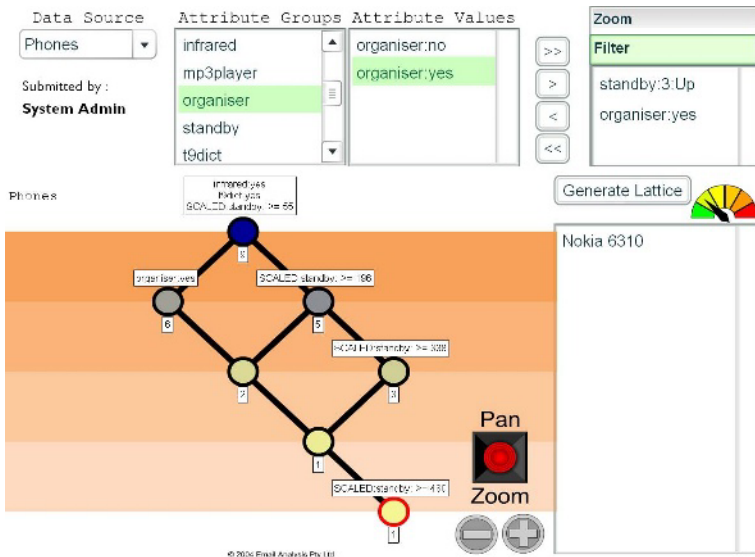
At this point the user could decide that infrared capability is the least desired feature and opt for the *Ericsson T65* as the phone to purchase.

#### 4.2 Case Scenario Two: Attribute Addition Method

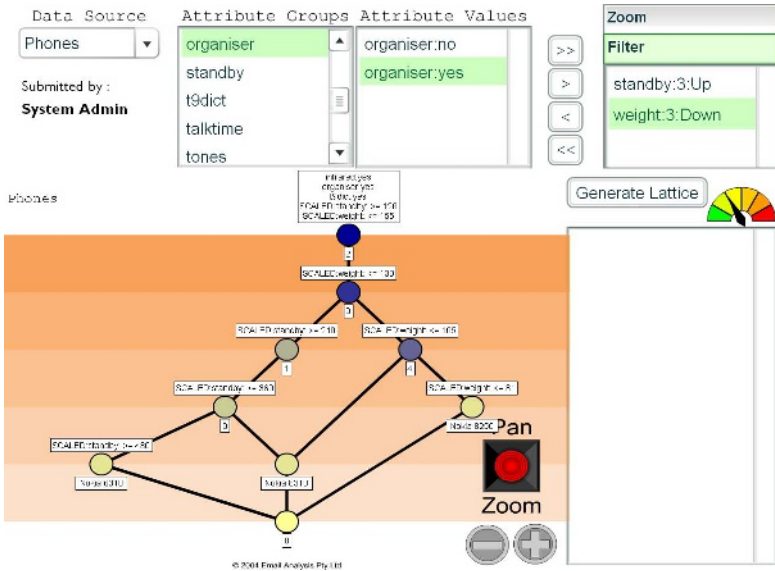
The user knows that two things he definitely wants in a cellphone are predictive text and infrared capabilities. He starts the search and adds `t9dict:yes` and `infrared:yes` as filter attributes.



This produces the simple lattice above showing 27 phones on the bottom concept. This means there are 27 phones with both predictive text and infrared capabilities. The list of 27 phones is too large for the user to reach a decision straight away so he promotes `t9dict:yes` and `infrared:yes` to zoom attributes. The user decides that an organiser and a long stand-by time are also important features which would influence his purchase decision, so adds the corresponding attributes as filters on the data. When adding stand-by time – the aim being to emphasis phones with a greater stand-by time – he configures the `stand-by` attribute to be ‘Ordinal Up’ resulting in the following diagram.



After looking at the generated lattice above, the user decides enough phones come with an organiser to warrant adding `organiser:yes` as a ‘zoom’ attribute. He realizes that a long stand-by time might come at the cost of increased phone weight. To ensure that the phone he gets is not too heavy for his needs, he adds `weight` with the order ‘Ordinal Down’ so that lighter phones are emphasized resulting the in following diagram.



The diagram above allows the user to quickly choose an optimum weight/stand-by time combination. It is easy to see in the above diagram that the phone most suitable to the requirements specified is the *Nokia 8310*.

## 5 Conclusion

At this point D-SIFT can perform the basic FCA operations against data quickly and dynamically. The final version of the user interface for the selection of mandatory attributes (*zooming*) is planned to be similar to TOSCANAJ where clicking a concept selects the concept’s intent as a restriction on the objects. This represents a minor implementation extension to the existing D-SIFT.

Furthermore, we are investigating the possibility of using the human input coded in conceptual scales from already existing FCA-based systems to support user search and data exploration. After parsing the standard storage documents from legacy FCA-based systems, D-SIFT could “offer” groups of attributes. Then the interaction starts from a given diagram, extending and changing it using the existing dynamic creation features of D-SIFT.

D-SIFT has recently been released to the Formal Concept Analysis community [3] where it is presented as a novel application of Formal Concept Analysis and a new workflow for FCA-based systems. However, in this presentation, and

to a data-warehousing audience, this paper emphasizes the combination of schema browsing for knowledge discovery using Formal Concept Analysis.

This paper has presented the architecture of the D-SIFT browser and illustrates the resulting D-SIFT-systems on two case scenarios against a database of cellular phones. The two examples demonstrate the generality of system integration outcomes from D-SIFT. The Conceptual Information Systems which result from applying the D-SIFT architecture present a new workflow for building and interacting with Formal Concept Analysis-based information systems. The workflow more closely aligns with dynamic schema interaction used in conceptual modeling.

**Acknowledgement.** This research work results from an International Linkage Grant supported by the ARC/DFG entitled *Conceptual Knowledge Processing*.

## References

1. P. Becker. Multi-dimensional Representations of Conceptual Hierarchies. In G. Mineau, editor, *Contributions to ICCS 2001, the 9th International Conference on Conceptual Structures*, pages 145–158, University Laval, 2001.
2. P. Becker, J. Hereth, and G. Stumme. ToscanaJ - An Open Source Tool for Qualitative Data Analysis. In *Advances in Formal Concept Analysis for Knowledge Discovery in Databases. Proc. Workshop FCAKDD of the 15th European Conference on Artificial Intelligence (ECAI 2002)*. Lyon, France., 2002.
3. J. Ducrou, B. Wormuth, and P. Eklund. D-SIFT: A Dynamic Simple Inuitive FCA tool. In M. Mugnier F. Dau and G. Stumme, editors, *13th Int. Conf. on Conceptual Structures*, LNCS 3596, pages 295–308, Heidelberg - Berlin - New York, 2005. Springer.
4. P. Eklund, J. Ducrou, and P. Brawn. Concept lattices for information visualization: Can novices read line diagrams. In P. Eklund, editor, *Proc. of the 2nd International Conference on Formal Concept Analysis - ICFCA'04*, pages 57–73. Springer, 2004.
5. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin, 1999.
6. J. Hereth Correira and T. Kaiser. A Mathematical Model for TOSCANA-Systems: Conceptual Data Systems. In P. Eklund, editor, *Concept Lattices*, pages 39–46, Heidelberg, Berlin, New York, 2004. Springer.

# FMC: An Approach for Privacy Preserving OLAP

Ming Hua, Shouzhi Zhang, Wei Wang, Haofeng Zhou, and Baile Shi

Fudan University, China

{minghua, shouzhi\_zhang, weiwang1, haofzhou, bshi}@fudan.edu.cn

**Abstract.** To preserve private information while providing thorough analysis is one of the significant issues in OLAP systems. One of the challenges in it is to prevent inferring the sensitive value through the more aggregated non-sensitive data. This paper presents a novel algorithm FMC to eliminate the inference problem by hiding additional data besides the sensitive information itself, and proves that this additional information is both necessary and sufficient. Thus, this approach could provide as much information as possible for users, as well as preserve the security. The strategy does not impact on the online performance of the OLAP system. Systematic analysis and experimental comparison are provided to show the effectiveness and feasibility of FMC.

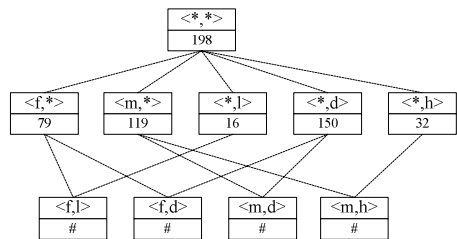
## 1 Introduction

Online analytic processing (OLAP) is an important infrastructure for advanced data analysis and knowledge discovery. While most of the previous studies on OLAP focus on OLAP models, data cube and data warehouse construction, maintenance and compression, as well as efficient query answering methods, it is critical to investigate the problem of *privacy preserving* in OLAP query answering.

**Example 1 (Motivation).** Consider a table about the patient cases in some hospitals as shown in Table 1.

**Table 1.** A table about the patient cases

Hospital	Disease	Number of cases
Forest	Lung cancer	16
Forest	Diabetes	63
Memorial	Diabetes	87
Memorial	Heart attack	32



**Fig. 1.** The data cubes based on Table 1

Suppose the hospitals do not want to make the population of individual diseases public, but agree to share the total number of all cases in a hospital or the total number of a certain disease in all hospitals. That is, in the data cube based on Table 1, the value

of cells  $\langle f,l \rangle$ ,  $\langle f,d \rangle$ ,  $\langle m,d \rangle$  and  $\langle m,h \rangle$  should be hidden from users (as shown in Figure 1.  $\langle f, l \rangle$  stands for the cell  $\langle \text{forest}, \text{lung cancer} \rangle$  and so do other cells).

A simple and direct security policy is to decline all the access to the sensitive cells. However, such a declining-direct-access policy is insufficient to preserve the privacy. Since just parts of the measure are hidden, the structure of the cube could be found out from the rest columns of the fact table, so the sensitive values could be revealed through other unprotected cells. For example, the value of  $\langle f,l \rangle$  is exactly the same as that of  $\langle *,l \rangle$ , since  $\langle *,l \rangle$  only aggregates this record. Moreover, subtracting the value of  $\langle f,l \rangle$  from that of  $\langle f,* \rangle$  discloses the value of  $\langle f,d \rangle$ .

Now, the problem becomes, “Can we make up a better security policy so that the privacy is strictly preserved?” Moreover, we want such a policy to hide as few information as possible. We call it the **privacy preserving OLAP problem**.

In this paper, we tackle the problem by hiding a minimal set of unprotected cells involved in determining the value of confidential cells, so that the precondition of information leakage will no longer hold. For example, if we hide the cells  $\langle *,l \rangle$  and  $\langle *,h \rangle$  in Figure 1, the value of the sensitive cells  $\langle f,l \rangle$ ,  $\langle f,d \rangle$ ,  $\langle m,l \rangle$  and  $\langle m,d \rangle$  will never be obtained by only accessing the remainder unprotected cells.

Compared to the privacy control problems in statistical database and data mining, there are several new challenges for the privacy preserving OLAP problem, and we make the following contributions.

1) Sensitive data items can be distributed at different granularity level in OLAP. *We propose a general model and solution that can handle this case.*

2) It is crucial for OLAP systems to provide users with as much information as possible while protecting the sensitive data. *We prove that our algorithm only hides the necessary data.*

3) OLAP applications usually require short response time. *We eliminate the inference before users interacting with the system*, so that the algorithm would not affect the online performance of the OLAP system.

The rest of the paper is organized as follows. In Section 2, we formulate the problem of privacy preserving OLAP. Then Section 3 provides the overview of the solution. The key techniques are discussed in section 4 and section 5. Extensive experimental results are reported in Section 6. Finally, we draw the conclusion in Section 7.

## 1.1 Related Work

Inference control methods in statistical databases are classified into two categories [1]. *Restriction based techniques* include auditing all queries [2], suppressing sensitive data [3] and so on. *Perturbation based techniques* include adding noise to source or outputs to affect the precision of detail data [4].

Inference control for OLAP systems received less attention. However, Lingyu Wang et al. have systematically studied this problem: [1] derives sufficient conditions for non-compromisability in sum-only data cubes; [5] discusses the inference problem caused by the multi-dimensional range queries; [6] proposes a method to eliminate both unauthorized accesses and malicious inferences.

## 2 Problem Definition

A *data cube* consists of a set of *dimensions* and *measures* with *aggregate functions* defined on it. In this paper, we mainly focus on the SUM function. Each node of the data cube is called a *cuboid*, and a tuple in the cuboid is called a *cell*. Two cuboids  $C_1$  and  $C_2$  follow the partial order (i.e.,  $C_1 \leq C_2$ ), iff on each dimension, either they share the same attribute, or  $C_2$  has a higher-level of attribute in the dimension hierarchy. In this case, we say  $C_2$  is an *ancestor* of  $C_1$ , and  $C_1$  is a *descendant* of  $C_2$ .  $C_2$  is a *father* of  $C_1$ , and correspondingly,  $C_1$  is a *son* of  $C_2$ , if  $C_1 \leq C_2$ , and there isn't any cuboid  $C$  such that  $C_1 \leq C$  and  $C \leq C_2$ . These definitions apply to cells as well. In Example 1, cuboids  $\langle \text{Hospital, Disease} \rangle \leq \langle \text{Hospital, *} \rangle$ , and the cells  $\langle f, l \rangle \leq \langle f, * \rangle$ .

Decided by the multi-dimensional data model, the access control in OLAP systems lies in *cuboids* and *cells*. We define the confidential information as a *forbidden set* in the form of  $\{c_1, \dots, c_m\}$ , where  $c_i$  is a cell of the data cube. We assume that the forbidden set includes all the confidential cells and their descendants, since a confidential cell could also be computed by simply aggregating all its descendants.

All the cells not included in the forbidden set compose the *available set*, which is accessible for users. For example, the available set in Example 1 includes all the cells except  $\langle f, l \rangle$ ,  $\langle f, d \rangle$ ,  $\langle m, d \rangle$  and  $\langle m, h \rangle$ . However, we have shown in Example 1 that some confidential information (such as  $\langle f, l \rangle$  and  $\langle f, d \rangle$ ) could be obtained by combining the cells in the available set. We define the available set as well as all the information derived from it as the *available set closure*.

**Definition 1 [Available Set Closure].** Given an available set  $A$ , the *Available Set Closure*  $C(A)$  is defined as:

1. If cell  $c \in A$ ,  $c \in C(A)$ ;
2. If cell  $c \in C(A)$ ,  $k \times c \in C(A)$ ,  $k$  is a real number;
3. If cells  $c_1, c_2 \in C(A)$ ,  $c_1 + c_2 \in C(A)$ ;

When the available set closure and the forbidden set have intersections, inference occurs. In this case, we also say that the forbidden set is *compromised*. The cells in the available set that cause the inference are called the *source* of the inference.

**Definition 2 [Compromisability].** Given a data cube  $L$  and a forbidden set  $F$  in  $L$ ,  $F$  is *compromised* when  $C(L - F) \cap F \neq \emptyset$ .

To prevent the compromisability, we hide some cells in the *source*, so that all the sensitive cells couldn't be computed through the incomplete source. However, the hidden cells may also be inferred by higher granular cells, therefore, more cells should be hidden to protect them. Finally we could find a set of cells in addition to the forbidden set, and any cell outside them would not cause inference to the cells inside.

**Definition 3 [Minimal Cover (MC)].** Given a data cube  $L$  and a Forbidden Set  $F$  in  $L$ , a set  $S$  is defined as the *Minimal Cover* of  $F$  (represented as  $MC(F)$ ) if:

1.  $S \subseteq L - F$ ;
2.  $C(L - F - S) \cap (F + S) = \emptyset$ .
3.  $\forall S' \subset S$ ,  $C(L - F - S') \cap (F + S') \neq \emptyset$



The minimal cover is a subset of the available set, and the second condition requires that after hiding the minimal cover, the remainder cells would not cause inference to both the minimal cover and the forbidden set. The third condition claims that any subset of the minimal cover couldn't satisfy the second one, which guarantees that all the cells in the minimal cover are indispensable to eliminate the inference.

**Problem Statement.** Given a data cube  $L$  and a forbidden set  $F$ , the **privacy preserving OLAP problem** is to find a minimal cover  $MC(F)$  of  $F$ , which prevents  $F$  from being compromised while prohibiting as few information as possible.

### 3 Overview of Privacy Preserving OLAP Procedure

From the definitions, it is clear that the minimal cover should be free of inference to both the forbidden set and itself; otherwise, one can disclose sensitive information by first inferring the values of minimal cover, and then getting to the forbidden set. A subset of the minimal cover that is only free of inference to the forbidden set is called the *minimal partial cover*.

We take the following two steps to firstly find the minimal partial cover of the forbidden set, and then extend it to the minimal cover to preserve absolute security.

**Step 1 Finding the minimal partial cover for the forbidden set.** We find the minimal partial cover MPC of the forbidden set by linear system theory, such that hiding MPC would eliminate all the inference direct to the forbidden set, but just hiding any subset of MPC would not work.

**Step 2 Extending the minimal partial cover to the minimal cover.** We then take MPC found in step 1 as the new forbidden set, and repeat finding the minimal partial cover for the newly hidden cells until no more cells need to be hidden.

### 4 Finding Minimal Partial Cover

In this section, we will discuss how to find the minimal partial cover for a forbidden set. First, we define the *vector code* to represent each cell in the cuboid as follows.

**Definition 5 [Vector Code].** Given a cuboid  $C$ , the vector code  $\bar{c}$  for cell  $c$  in  $C$  or  $C$ 's father cuboids is defined as  $(a_1, \dots, a_n)$ , where  $n$  is the number of cells in  $C$ , and

$$a_i = \begin{cases} 1 & \text{if } c \text{ is the } i\text{th cell in } C \text{ (} c \in C \text{) or } a_i = \begin{cases} 1 & \text{if } c \text{ aggregates the } i\text{th cell in } C \text{ (} c \in \text{Father}(C) \text{).} \\ 0 & \text{otherwise} \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

For example, in the cuboid <Hospital, Disease> in Figure 1, the vector code of cell <f,\*> is (1,1,0,0), and the vector for <f, l> is (1,0,0,0). The cell corresponding to  $\bar{c}$  could be inferred by  $c_1, \dots, c_n$ , if vector codes  $\bar{c}_1, \dots, \bar{c}_n$  can be linearly combined into the vector code  $\bar{c}$ . To determine whether it would happen, we discuss the following three cases of the solution of equation (1): ( $x_1, \dots, x_n$  are real numbers).

$$x_1 \times \bar{c}_1 + \dots + x_n \times \bar{c}_n = [\bar{c}_1, \dots, \bar{c}_n] \times [x_1, \dots, x_n]^T = \bar{c} \tag{1}$$

- **Equation (1) has no solutions.** Cell  $c$  corresponding to  $\bar{c}$  couldn't be computed with any other cells, so no additional information needs to be hidden.
- **Equation (1) has only one non-zero solution.**  $\bar{c}$  could be computed with a certain combination of  $\bar{c}_1, \dots, \bar{c}_n$ . If  $x_i, \dots, x_j$  are the non-zero components of the solution, then the corresponding cells  $\bar{c}_i, \dots, \bar{c}_j$  are indispensable to inferring  $\bar{c}$ . Therefore, just hiding one of  $\bar{c}_i, \dots, \bar{c}_j$  could prevent the inference.
- **Equation (1) has more than one non-zero solutions.** To eliminate all the inference, we need to hide one cell whose corresponding component of solution  $X$  is always non-zero. If there isn't such kind of cells, we need to find a set of cells at least one of which is used in each solution.

### 4.1 An Example

Based on linear system theory [7], we develop a method to eliminate the inference to certain cells. The method is illustrated in the following example.

**Example 2.** We try to find the minimal partial cover for cell  $\langle f,d \rangle$  in Example 1, and the security requirements are the same. Suppose  $c_1 = \langle f,* \rangle$ ,  $c_2 = \langle m,* \rangle$ ,  $c_3 = \langle *,l \rangle$ ,  $c_4 = \langle *,d \rangle$ , and  $c_5 = \langle *,h \rangle$ . The corresponding vectors are  $\bar{c}_1, \dots, \bar{c}_5$ .

1. We construct the equation by making  $A = [\bar{c}_1, \dots, \bar{c}_5]$ ,  $b = \bar{c}$  (vector code of  $\langle f,d \rangle$ ).

$$AX = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \times X = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

2. The solution of equation (2) is  $X = X_0 + k \times X_1$ , where  $X_0 = [1, 0, -1, 0, 0]^T$ ,  $X_1 = [-1, -1, 1, 1, 1]^T$ , and  $k$  is a real number. If the  $i_{th}$  component of  $X$  is non-zero, then  $\bar{c}_i$  is used to compute  $\langle f,d \rangle$ . For example, if we take  $k=0$ , then  $X = [1, 0, -1, 0, 0]^T$ , (i.e.,  $\bar{c} = \bar{c}_1 - \bar{c}_3$ ), which is exactly the case depicted in Example 1.

---

**Input:** The forbidden set  $F$ , and the cuboid  $C$

**Output:** A minimal partial cover  $MPC$  of  $F$

**Method:**

- 1: construct the coefficient matrix  $A = [\bar{c}_1 \dots \bar{c}_n]$
  - 2: for each cell  $c$  in  $F$
  - 3: if  $Ax = \bar{c}$  has solutions
  - 4: find the solutions  $X$  of  $Ax = \bar{c}$
  - 5: find the set of components  $M_c$  at least one of which is non-zero in each  $X$
  - 7: return  $MPC = \bigcup_{c \in F} M_c$
- 

**Fig. 2.** Algorithm 1 FMPC: finding a minimal partial cover

3. We try to find a component of X that is always non-zero, or find a set of components at least one of which is non-zero in each X.
  - **If k=0:** X=X<sub>0</sub>, the first and third components are non-zero.
  - **If k≠0:** by carefully choosing a value for k, the first or the third component can be zero, but the other components will never be zero. Hence, a cell in {c<sub>1</sub>, c<sub>3</sub>} and another one in {c<sub>2</sub>, c<sub>4</sub>, c<sub>5</sub>} form the minimal partial cover of <f,d>. For example, if we hide {c<sub>1</sub>, c<sub>5</sub>}, <f,d> wouldn't be compromised.

### 4.2 Algorithm

Now, let us generalize the algorithm of finding the minimal partial cover (Figure 2). Given a forbidden set F in cuboid C, first construct the coefficient matrix A using the unprotected cells in C or C's fathers. Then for each cell c in F, if Ax=c has solutions, find the set of components in the solutions at least one of which is non-zero in each X.

Here we use linear system theory [7] to find such cells. The solutions of Ax=c can be represented as  $x=x_0+[x_1, \dots, x_r] \times [k_1, \dots, k_r]$ , where  $x_1, \dots, x_r$  is the basic solutions of Ax=0, and X<sub>0</sub> is a certain solution of Ax=c. There are r "independent" components in X, taking zero in x<sub>0</sub> and taking "1" respectively in each x<sub>i</sub> (i=1, ..., r). For example, in figure 3, the last three components are independent. Suppose X<sub>0</sub>[i] and X<sub>2</sub>[i] are non-zero in all the i<sup>th</sup> components of X<sub>0</sub> to X<sub>3</sub>, and X<sub>2</sub>[j] is the independent component taking "1" in X<sub>2</sub>, then either X[i] or X[j] is used in X, and the corresponding cells are the minimal partial cover.

$$\begin{array}{c}
 \begin{matrix} X \\ \# \\ X[i] \\ \# \\ \# \\ X[j] \\ \# \end{matrix}
 \end{array}
 =
 \begin{array}{c}
 \begin{matrix} X_0 \\ \# \\ X_0[i] \\ \# \\ 0 \\ 0 \\ 0 \end{matrix}
 \end{array}
 +
 \begin{array}{c}
 \begin{matrix} k_1 \\ k_2 \\ k_3 \end{matrix}
 \end{array}
 \times
 \begin{array}{c}
 \begin{matrix} X_1 & X_2 & X_3 \\ \# & \# & \# \\ 0 & X_2[i] & 0 \\ \# & \# & \# \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}
 \end{array}
 \left. \vphantom{\begin{matrix} X \\ \# \\ X[i] \\ \# \\ \# \\ X[j] \\ \# \end{matrix}} \right\} \begin{array}{l} \text{Independent} \\ \text{Components} \end{array}$$

Fig. 3. An example of minimal partial cover

**Lemma 1.** Given  $X=X_0+[X_1, \dots, X_{n-r}] \times [k_1, \dots, k_{n-r}]^T$ , the (r+1)<sup>th</sup> to n<sup>th</sup> component of X are the independent components. If X<sub>0</sub>[i]≠0, and only X<sub>d<sub>1</sub></sub>[i], ..., X<sub>d<sub>j</sub></sub>[i] of X<sub>1</sub>[i], ..., X<sub>n-r</sub>[i] are non-zero (d<sub>1</sub>, ..., d<sub>j</sub> ∈ {1, ..., n-r} and i<r+1), then:

1. At least one of the components X[i], X[r+d<sub>1</sub>], ..., X[r+d<sub>j</sub>] in X would be non-zero.
2. Any subset of components X[i], X[r+d<sub>1</sub>], ..., X[r+d<sub>j</sub>] could all be zero in X.

**Lemma 2.** Algorithm 1 returns a minimal partial cover of the forbidden set FS. (The proof of Lemma 1 and Lemma 2 are not provided here due to the limit of space.)

## 5 Extending the Minimal Partial Cover to Minimal Cover

In this section, we employ a level-wise framework to extend the minimal partial cover to the minimal cover to each cuboid of the cube with some optimizing strategies.

### 5.1 Two Optimizing Strategies

**Eliminating Single-son Inference.** A cell is called a *single-son* cell if it has only one child in its son cuboid. All the single-son fathers of the forbidden set are definitely sensitive. In Example 1, if we hide the two single-son cell  $\langle *,l \rangle$  and  $\langle *,h \rangle$ , all inferences will be eliminated. Thus, in our algorithm we first add all the single-son fathers of the sensitive cells to the minimal cover. It may both eliminate a large part of inference and reduce the number of cells we must check for inference.

**Finding Candidate Range.** In algorithm 1, we check all the fathers and unprotected siblings of the forbidden cells for inference. However, not all of them are dangerous.

**Example 3.** A two-dimensional cube is shown in Figure 4(a). The cell  $\langle a_2, b_1 \rangle$  marked with “\*” in the cuboid  $\langle A, B \rangle$  is sensitive.

We construct the coefficient matrix A for cuboid  $\langle A, B \rangle$  (as shown in Figure 4(b)). The column vectors of A are related with 8 father cells and 5 unprotected cells in cuboid  $\langle A, B \rangle$ . However, only the column vector A[1], A[2], A[5], A[6], A[9] and A[10] are probable to infer the value of  $\langle a_2, b_1 \rangle$ , because others have all zeros in the corresponding components. We call the sub matrix formed by A[1], A[2], A[5], A[6], A[9], A[10] and the non-zero components of them the *candidate range* of the forbidden set (surrounded with dashed in Figure 4(b)). The candidate range could be found by first setting it to the father cells of the forbidden set, and then iteratively add in the cells which intersect with the candidate range.

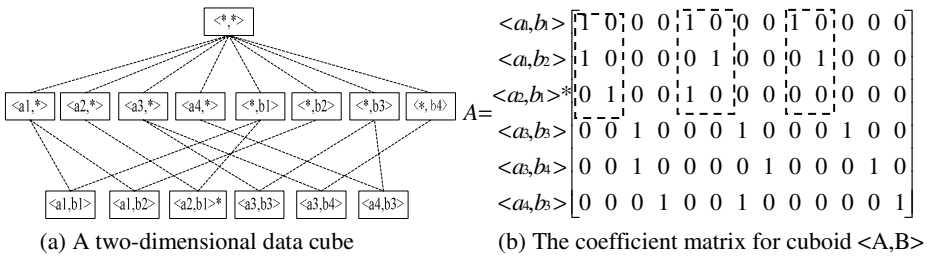


Fig. 4. A two-dimensional data cube

### 5.2 Algorithm

We use a level-wise framework to extend the minimal partial cover to minimal cover. As shown in the Algorithm 2 in Figure 5, we first rank the cuboids in the cube according to the ascend order of the granularity level. Then, for each cuboid, we apply the two optimizing strategies, and invoke Algorithm 1 to find the minimal partial cover of the forbidden set in this cuboid. The returned minimal partial cover should be further checked for inference. This process should be repeated until there isn't any new minimal partial cover in the current cuboid.

---

**Input:** The forbidden set FS  
**Output:** A minimal cover MC of FS  
**Method:**

- 1: for each cuboid  $C^*$  in the cube
- 2:     while  $FS \cap C^* \neq \emptyset$
- 3:         Add single son father to MC
- 4:         find the candidate range CR for FS
- 5:          $m = FMPC(FS, CR)$   
           //m is the minimal partial cover of FS returned by FMPC
- 6:          $FS = FS - FS \cap C^*$      //inference to  $FS \cap C^*$  has been eliminated
- 7:          $MC = MC \cup m$
- 8:          $FS = FS \cup m$      //the minimal partial cover should be protected
- 9: return MC

---

**Fig. 5.** Algorithm 2 (FMC: a level wise algorithm to find a minimal cover)

**Theorem 1.** Algorithm 2 returns a minimal cover of the forbidden set  $FS$ .  
 (The proof of Theorem 1 is based on Lemma 1 and Lemma 2, and is not provided here due to the limit of space.)

## 6 Experimental Results

**Implementation.** All experiments are conducted on a Pentium4 2.80 GHz PC with 512MB main memory, running Microsoft Windows XP Professional. The algorithm is implemented using Borland C++ Builder 6 with Microsoft SQL Server 2000.

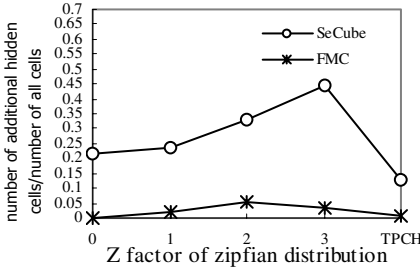
**Data Set.** We used the synthetic data sets and real data set TPC-H benchmark for our experiments. In synthetic data sets, we generated data from a Zipfian distribution<sup>1</sup>, skew of the data ( $z$ ) was varied over 0, 1, 2 and 3. The sizes of the data sets vary from 20000 to 80000 cells, with 3 dimensions and 4 granularity levels in one dimension.

**Comparison on Different Zipf Parameter.** We apply FMC to TPC-H benchmark and the synthetic datasets whose parameter  $z=0, 1, 2,$  and  $3$ . We randomly select 1% of the cells in two cuboids as the forbidden set, and compared the additional cells hidden by FMC and SeCube (L. Wang et al. 2004). Figure 6(a) shows the results. When  $z=0$ , the data is uniformly distributed, fewer additional cells need to be hidden than that in the skewed case. Because some values of the dimension appear less often in the skewed dataset, these “sparse” data are the main cause of inference.

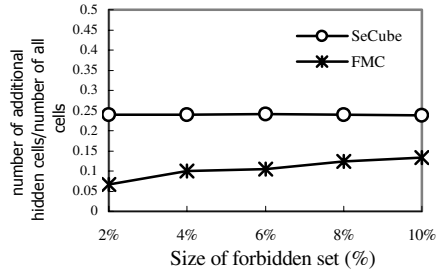
We also evaluate the effectiveness of the two optimizing strategies. Figure 7(a) with the size of candidate range shows that at most 50% of the cube needs to be check for inference. Figure 7(b) shows the number of single-son inference cases. Since it takes a significant part in all inference cases, to eliminate the single-son inference first will contribute to the approach greatly.

---

<sup>1</sup> The generator is obtained via [ftp.research.microsoft.com/users/viveknar/tpcdskew](http://ftp.research.microsoft.com/users/viveknar/tpcdskew)

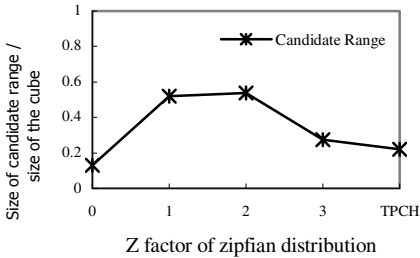


(a) Compare on different zipf factors

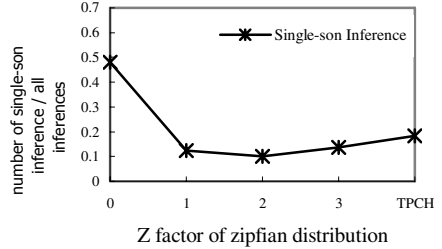


(b) Compare on different forbidden set size

**Fig. 6.** Size of additional protected cells / size of cube

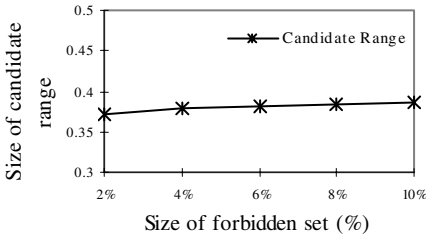


(a) Size of candidate range/size of cube

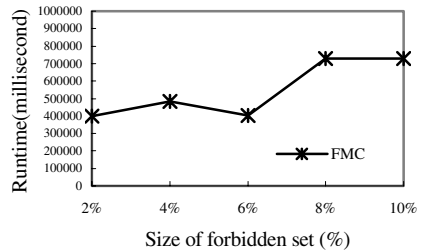


(b) single-son inference/all inference cases

**Fig. 7.** Experimental result of two optimizing strategies



**Fig. 8.** Size of candidate range / size of cube



**Fig. 9.** Runtime of FMC

**Comparison on Varied Forbidden Set.** We set the zipf parameter to  $z=1$ , and change the size of forbidden set. Figure 6(b) shows the size of additional cells hidden by SeCube [6] and FMC, where FMC hide fewer cells than SeCube in all cases.

Figure 8 demonstrates the candidate range on different forbidden set size. The size of candidate range stays below 40% in all cases, which means that we only need to check 40% of the whole cube for inference. We also tested the runtime of FMC for different size of forbidden set (Figure 9).

## 7 Conclusions

In this paper, we present an effective and efficient algorithm to address the privacy preserving OLAP problem. The main idea is to hide part of the data causing the inference, so that the sensitive information could no longer be computed. We could guarantee that all the information we hide is necessary, and thus as much information as possible can be provided for users while protecting the sensitive data. All work will be done before users interacting with the system, and thus, it would not affect the online performance of the OLAP system. Our algorithm is partially based on the linear system theory, so the correctness could be strictly proved. Experimental results also demonstrate the effectiveness of the algorithm. Future work includes applying the method to other aggregation functions and improving the efficiency of the algorithm. We also plan to extend the work to solve the inference problem caused by involving two aggregation functions in one cube.

## References

1. L. Wang, D. Wijesekera: Cardinality-based Inference Control in Sum-only Data Cubes. Proc. of the 7th European Symp. on Research in Computer Security, 2002.
2. F. Y. Chin, G. Ozsoyoglu: Auditing and inference control in statistical databases. IEEE Trans. on Software. Eng. pp. 574-582 (Apr. 1982)
3. L.H. Cox: Suppression methodology and statistical disclosure control. Journal of American Statistic Association, 75(370):377-385, 1980.
4. D. E. Denning: Secure statistical databases under random sample queries. ACM Trans. on Database Syst. Vol. 5(3) pp. 291-315 (Sept. 1980)
5. L. Wang, Y. Li, D. Wijesekera, S. Jajodia: Precisely Answering Multi-dimensional Range Queries without Privacy Breaches. ESORICS 2003: 100-115
6. L. Wang, S. Jajodia, D. Wijesekera: Securing OLAP data cubes against privacy breaches. Proc. IEEE Symp. on Security and Privacy, 2004, pages 161-175.
7. K. Nicholson: Elementary Linear Algebra. Second Edition, McGraw Hill, 2004.

# Information Driven Evaluation of Data Hiding Algorithms

Elisa Bertino<sup>1</sup> and Igor Nai Fovino<sup>2</sup>

<sup>1</sup> CERIAS and CS Department, Purdue University  
bertino@cerias.purdue.edu

<sup>2</sup> Institute for the Protection and the Security of the Citizen,  
Joint Research Centre  
igor.nai@jrc.it

**Abstract.** Privacy is one of the most important properties an information system must satisfy. A relatively new trend shows that classical access control techniques are not sufficient to guarantee privacy when datamining techniques are used. Privacy Preserving Data Mining (PPDM) algorithms have been recently introduced with the aim of modifying the database in such a way to prevent the discovery of sensible information. Due to the large amount of possible techniques that can be used to achieve this goal, it is necessary to provide some standard evaluation metrics to determine the best algorithms for a specific application or context. Currently, however, there is no common set of parameters that can be used for this purpose. This paper explores the problem of PPDM algorithm evaluation, starting from the key goal of preserving of data quality. To achieve such goal, we propose a formal definition of data quality specifically tailored for use in the context of PPDM algorithms, a set of evaluation parameters and an evaluation algorithm. The resulting evaluation core process is then presented as a part of a more general three step evaluation framework, taking also into account other aspects of the algorithm evaluation such as efficiency, scalability and level of privacy.

## 1 Introduction

Intense work in the area of data mining technology and in its applications to several domains has resulted in the development of a large variety of techniques and tools able to automatically and intelligently transform large amounts of data in knowledge relevant to users. The use of these tools is today the only effective way to extract useful knowledge from the increasing number of very large databases, that, because of their sizes, cannot be manually analyzed. However, as with other kinds of useful technologies, the knowledge discovery process can be misused. It can be used for example by malicious subjects to reconstruct sensitive information for which they do not have access authorization. Because of the nature of datamining techniques that use non sensitive data in order to infer hidden information, the usual security techniques are actually not able to prevent illegal accesses carried out through the use of data mining techniques. For this reason, many research efforts have been recently devoted to addressing



the problem of privacy preserving in data mining. As a result, different sanitization techniques have been proposed for hiding sensitive items or patterns by removing some of the data to be released or inserting noise into data. Because, however, the sets of parameters adopted for assessing the various algorithms are very heterogeneous, it is difficult to compare these algorithms in order to determine which is the most suitable one for a given context or application. Moreover all parameters adopted by the various metrics do not take into account an important issue of the privacy-preserving data mining (PPDM) process, that is, the quality of the data obtained as result from the sanitization process. In this paper, we explore the DQ properties that are relevant for the evaluation of PPDM algorithms. We propose a model to describe aggregated information, their constraints and their relevance in order to evaluate the various DQ parameters and we provide a three-step framework to evaluate PPDM algorithms using DQ as final discriminant.

## 2 Related Work

The first approach dealing with the problem of DQ for perturbed data has been developed by Agrawal and Srikant [6]. More in detail, they propose an approach to estimate the privacy level introduced by their PPDM algorithm. Such an approach evaluates the accuracy with which the original values of a modified attribute can be determined. A similar approach has been developed by Rivzi and Haritsa [8]. They propose a distortion method to pre-process the data before executing the mining process. The privacy measure they propose deals with the probability with which the distorted entries can be reconstructed.

These initial approaches have then been extended by taking into account other evaluation parameters and by considering specific data mining techniques. In particular, Agrawal and Aggarwal [5] analyze the proposed PPDM algorithm with respect to two parameters, that is, privacy and information loss. In the context of clustering techniques, Oliveira and Zaiane in [3][2] define some interesting parameters to evaluate the proposed PPDM algorithms, each concerning different aspects of these algorithms. In particular, in the context of Data quality, interesting parameters they introduced are the *misclassification error*, used to estimate how many legitimate data points are incorrectly classified in the distorted database and the *Artifactual Pattern* estimating the artifactual patterns introduced by the sanitization process that are not present in the original database. A more comprehensive evaluation framework for PPDM algorithms has been recently developed by Bertino et al. [16]. Such an evaluation framework consists of five parameters: the *Efficiency*, the *Scalability*, the *Data quality*, the *Hiding failure* and the *Level of privacy*. As it can be noticed from the above overview, only the approaches by Bertino et al. [16] and Oliveira and Zaiane [2,3] have addressed the problem of DQ in the context of the PPDM process. However, these approaches have some major differences with respect to the work presented in this paper. As we discussed before, in the evaluation framework developed by Bertino et al., DQ is directly measured by the dissimilarity pa-

parameter. By contrast, Olivera and Zaiane only measure DQ through the estimation of the artifactual patterns and the misclassification error. However, none of these approaches completely captures the concept of DQ. In particular, there are some aspects related to DQ evaluation that are heavily related not only with the PPDM algorithm, but also with the structure of the database, and with the meaning and relevance of the information stored in the database with respect to a well defined context. Parameters such as dissimilarity, artifactual information and misclassification error are not able to capture these aspects. The goal of this paper is to explore the role and relevance of DQ in the PPDM process by developing a more appropriate set of instruments to assess the quality of the data obtained by the sanitization process.

### 3 Data Quality in the Context of PPDM

Traditionally DQ is a measure of the consistency between the data views presented by an information system and the same data in the real-world [10]. This definition is strongly related with the classical definition of information system as a “model of a finite subset of the real world” [12]. More in detail Levitin and Redman [13] claim that DQ is the instrument by which it is possible to evaluate if data models are well defined and data values accurate. The main problem with DQ is that its evaluation is relative [9], in that it usually depends from the context in which data are used. In the scientific literature DQ is considered a multi-dimensional concept that in some environments involves both objective and subjective parameters [11,14]. In the context of PPDM, we are interested in assessing whether, given a target database, the sanitization phase will compromise the quality of the mining results that can be obtained from the sanitized database. The parameters we consider relevant in the context of PPPDM are the following: the *Accuracy*, measuring the proximity of a sanitized value  $a^I$  to the original value  $a$ ; the *Completeness*, evaluating the percentage of data from the original database that are missing from the sanitized database and finally the *Consistency* that is related to the semantic constraints holding on the data and it measures how many of these constraints are still satisfied after the sanitization. We now present the formal definitions of those parameters for use in the remainder of the discussion. Let  $OD$  be the original database and  $SD$  be the sanitized database resulting from the application of the PPDM algorithm. Without losing generality and in order to make simpler the following definitions, we assume that  $OD$  (and consequently  $SD$ ) be composed by a single relation. We also adopt the positional notation to denote attributes in relations. Thus, let  $od_i$  ( $sd_i$ ) be the  $i$ -th tuple in  $OD$  ( $SD$ ), then  $od_{ik}$  ( $sd_{ik}$ ) denotes the  $k^{th}$  attribute of  $od_i$  ( $sd_i$ ). Moreover, let  $n$  be the total number of the attributes of interest, we assume that attributes of positions  $1, \dots, m$  ( $m \leq n$ ) are the primary key attributes of the relation.

**Definition 1:** Let  $sd_j$  be a tuple of  $SD$ . We say that  $sd_j$  is **Accurate** if  $\neg \exists od_i \in OD$  such that  $((od_{ik} = sd_{jk}) \forall k = 1..m \wedge \exists (od_{if} \neq sd_{jf}), (sd_{jf} \neq NULL), f = m + 1, \dots, n)$ .  $\square$

**Definition 2:** A  $sd_j$  is **Complete** if  $(\exists od_i \in OD \text{ such that } (od_{ik} = sd_{jk}) \forall k = 1..m) \wedge (\neg \exists (sd_{jf} = NULL), f = m + 1, \dots, n)$ .  $\square$

Let  $C$  the set of the constraints defined on database  $OD$ , in what follows we denote with  $c_{ij}$  the  $j^{\text{th}}$  constraint on attribute  $i$ . We assume here constraints on a single attribute, but, as we show in Section 4 it is easily possible to extend the measure to complex constraints.

**Definition 3:** An instance  $sd_k$  is **Consistent** if  $\neg \exists c_{ij} \in C \text{ such that } c_{ij}(sd_{ki}) = false, i = 1..n$   $\square$

## 4 Information Driven Data Quality Schema

Current approaches to PPDM algorithms do not take into account two important aspects:

- **Relevance of data:** not all the information stored in the database has the same level of relevance and not all the information can be dealt at the same way.
- **Structure of the database:** information stored in a database is strongly influenced by the relationships between the different data items. These relationships are not always explicit.

We believe that in a context in which a database administrator needs to choose which is the most suitable PPDM algorithm for a target real database, it is necessary to also take into account the above aspects. To achieve this goal we propose to use Data Quality in order to assess how and if these aspects are preserved after a data hiding sanitization.

### 4.1 The Information Quality Model

In order evaluate DQ it is necessary to provide a formal description that allow us to magnify the aggregate information of interest for a target database and the relevance of DQ properties for each aggregate information (AI) and for each attribute involved in the AI. The Information Quality Model (IQM) proposed here addresses this requirement. In the following, we give a formal definition for an Attribute Class (AC), a Data Model Graph (DMG) (used to represent the attributes involved in an aggregate information and their constraints) and an Aggregation Information Schema (AIS). Before giving the definition of DMG, AIS and ASSET we introduce some preliminary concepts.

**Definition 4:** An Attribute Class is defined as the tuple  $AT_C = \langle name, AW, AV, CW, CV, CSV, Slink \rangle$  where:

- *Name* is the attribute id
- *AW* is the accuracy weigh for the target attribute
- *AV* is the accuracy value

- $CW$  is the completeness weigh for the target attribute
- $CV$  is the completeness value
- $CSV$  is the consistency value
- $Slink$  is list of simple constraints. □

**Definition 5:** A Simple Constraint Class is defined as the tuple  $SC_C = \langle name, Constr, CW, Clink, CSV \rangle$  where:

- $Name$  is the constraint id
- $Constraint$  describes the constraint using some logic expression
- $CW$  is the weigh of the constraint. It represents the relevance of this constraint in the  $AIS$
- $CSV$  is the number of violations to the constraint
- $Clink$  it is the list of complex constraints defined on  $SC_C$ . □

**Definition 6:** A Complex Constraint Class is defined as the tuple  $CC_C = \langle name, Operator, CW, CSV, SC_Clink \rangle$  where:

- $Name$  is the Complex Constraint id
- $Operator$  is the “Merging” operator by which the simple constraints are used to build the complex one.
- $CW$  is the weigh of the complex constraint
- $CSV$  is the number of violations
- $SC_Clink$  is the list of all the  $SC_C$  that are related to the  $CC_C$ . □

Let  $D$  a database, we are able now to define the DMG, AIS and ASSET on  $D$ .

**Definition 7:** A DMG (Data Model Graph) is an oriented graph with the following features:

- A set of nodes  $N_A$  where each node is an Attribute Class
- A set of nodes  $SC_C$  where each node describes a Simple Constraint Class
- A set of nodes  $CC_C$  where each node describes a Complex Constraint Class
- A set of direct edges  $L_{N_j, N_k} : L_{N_j, N_k} \in ((N_A X SC_C) \cup (SC_C X CC_C) \cup (SC_C X N_A) \cup (CC_C X N_A))$ . □

**Definition 8:** An AIS  $\phi$  is defined as a tuple  $\langle \gamma, \xi, \lambda, \vartheta, \varpi, W_{AIS} \rangle$  where:  $\gamma$  is a name,  $\xi$  is a DMG,  $\lambda$  is the accuracy of  $AIS$ ,  $\vartheta$  is the completeness of  $AIS$ ,  $\varpi$  is the consistency of  $AIS$  and  $W_{AIS}$  represent the relevance of  $AIS$  in the database. □

We are now able to identify as **ASSET** (Aggregate information Schema Set) as the collection of all the relevant  $AIS$  of the database.

The DMG completely describes the relations between the different data items of a given AIS and the relevance of each of these data respect to the data quality parameter. It is the “road map” that is used to evaluate the quality of a sanitized AIS.

## 4.2 Data Quality Evaluation of AIS

By adopting the IQM scheme, now we are able to evaluate the data quality at the attribute level. By recalling *Definition (1,2)*, we define the *Accuracy lack* of

an attribute  $k$  for an AIS  $A$  as the proportion of non accurate items in a database  $SD$ . Ate the same way, the *Completeness lack* of an attribute  $k$  is defined as the proportion of non complete items in  $SD$ . The accuracy lack index for an AIS can the be evaluated as follows:

$$ACL = \sum_{i=0}^{i=n} DMG.N_i.AV * DMG.N_i.AW \quad (1)$$

where  $DMG.N_i.AW$  is the accuracy weight associated with the attribute identified by the node  $N_i$ . Similarly the completeness lack of an AIS can be measured as follows:

$$CML = \sum_{i=0}^{i=n} DMG.N_i.CV * DMG.N_i.CW \quad (2)$$

Finally the consistency lack index associated with an AIS is given by number of constraint violations occurred in all the sanitized transaction multiplied by the weight associated with every constraints (simple or complex).

$$CSL = \sum_{i=0}^{i=n} DMG.SC_i.csv * DMG.SC_i.cw + \sum_{j=0}^{j=m} DMG.CC_j.csv * DMG.CC_j.cw \quad (3)$$

### 4.3 The Evaluation Algorithm

In this section we present the methodology we have developed to evaluate the data quality of the AIS. This methodology is organized in two main phases:

- **Search:** in this phase all the tuples modified in the sanitized database are identified. The primary keys of all these transaction (we assume that the sanitization process does not change the primary key), are stored in a set named *evalset*. This set is the input of the Evaluation phase.
- **Evaluation:** in this phase the accuracy, the consistency and the completeness associated with the DMG and the AIS are evaluated using information on the accuracy and completeness weight associated with the DMG and related to the transactions in Evalset.

The algorithms for these phases are reported in Figures 1 and 2. Once the evaluation process is completed, a set of values is associated with each AIS that gives the balanced level of accuracy, completeness and consistency. However, this set may not be enough. A simple average of the different AIS's values could not be significant, because even in this case not all the AIS's in the ASSET have the same relevance. For this reason, a weight is associated with each AIS that represents the importance of the high level information represented by the AIS in the target context. The accuracy, the completeness and the consistency of the ASSET for each PPDM algorithm candidate are then evaluated as follows:

$$Accuracy_{Asset} = \frac{\sum_{i=0}^{i=|Asset|} AIS_i.accuracy * AIS_i.W}{|Asset|} \quad (4)$$

---

```

INPUT: Original database OD, Sanitized database SD
OUTPUT: a set Evalset of primary keys
Begin
  ForEach  $t_i \in OD$  do
    {j=0;
     While ( $s_{jk} \neq t_{ik}$ )and( $j < |SD|$ )do j ++;
     l=0;
     While ( $s_{jl} = t_{il}$ )and( $l < n$ ) do l++;
     If( $l < n$ )Then Evalset = Evalset  $\cup$   $t_{ik}$ 
    }
End

```

---

Fig. 1. Search algorithm

---

```

INPUT: the original database OD, the sanitized database SD, Evalset, IQM.
OUTPUT: the IQM containing a data quality evaluation
Begin
  ForEach IES in IQM do
    {DMG = IQM.IES.link;
     avet = 0; cvet = 0;
     ForEach ( $t_{ik} \in Evalset$ )do
       For(j = 0; j < n; j ++)do
         If ( $t_{ij} \neq s_{ij}$ ) Then
           {
             If  $s_{ij} = NULL$  Then cvet[j] ++;
             Else avet[j] ++;
             validate_constr(IES, DMG, j)
           }
       For(m = 0; m < n; m ++)do
         { DMG.Nm.AV =  $\frac{avet[m]}{|SD|}$ ; DMG.Nm.CV =  $\frac{cvet[m]}{|SD|}$ ;
           IQM.IES.AV =  $\sum_{i=0}^{i=n} (DMG.N_i.AV * DMG.N_i.AW)$ ;
           IQM.IES.CV =  $\sum_{i=0}^{i=n} (DMG.N_i.CV * DMG.N_i.CW)$ ;
           IQM.IES.CSV =  $\sum_{i=0}^{i=n} \sum_{j=0}^{j=m} DMG.SC_i.CSV * DMG.SC_i.CW +$ 
              $\sum_{j=0}^{j=m} DMG.CC_j.CSV * DMG.CC_j.CW$ ; }
     }
End
Procedure validate_constr(IES, DMG, j) Begin
   $N_A = AIS.DMG.j$ 
  For k=1; k <  $|N_A.stink|$ ; k++
  {  $N_C = N_A.Stink[k]$ 
    if  $N_C.Clink == NULL$  then if  $!(N_C.constr(s_{ij}))$  then  $N_C.CSV ++$ ;
    else {  $N_O = N_C.Clink$ ; globalconstr = composeconstr( $N_O, s_{ij}$ )
      if  $!(globalconstr)$  then  $N_O.CSV ++$  }
  }
End

```

---

Fig. 2. Evaluation Algorithm

$$Completeness_{Asset} = \frac{\sum_{i=0}^{i=|Asset|} AIS_i.completeness * AIS_i.W}{|Asset|} \quad (5)$$

$$Consistency_{Asset} = \frac{\sum_{i=0}^{i=|Asset|} AIS_i.consistency * AIS_i.W}{|Asset|} \quad (6)$$

where  $AIS_i.W$  represents the weight (relevance) associated with the  $i$ -th AIS.

## 5 Evaluation Framework

As shown by the approaches reported in Section 2, and especially by the approach by Bertino et al. [16], in many real world applications, it is necessary to take into account even other parameters that are not directly related to DQ. On the other hand we believe that DQ should represent the invariant of a PPDM

evaluation and should be used to identify the best algorithm within a set of previously selected “Best Algorithms”. To preselect this “best set”, we suggest to use some parameters as discriminant to select the algorithms that have an acceptable behavior under some aspects generally considered relevant especially in “production environments” (efficiency, scalability, hiding failure and level of privacy). In order to understand if these four parameters are sufficient to identify an acceptable set of candidates, we performed an evaluation test. We identified a starting set of PPDM algorithms for Association Rules Hiding (the algorithms presented in [4] and a new set of three algorithms based on data fuzzification [16]). Then, by using the IBM synthetic data generator<sup>3</sup> we generated a categorical database representing an hypothetical Health Database storing the different therapies associated with the patients. We also built the associated DMG. On this database, we applied the different algorithms and then we measured the previous parameters. Once we built the “Best Set” we discovered that some algorithms that performed less changes to the database, which in some way indicates a better quality, are not in this set. A reason is for example a low efficiency. For this reason we believe that even in the preselection phase a “coarse” DQ parameter must be introduced. In our opinion, the *Coarse DQ Measure* depends on the specific class of PPDM algorithms. If the algorithms adopt a perturbation or a blocking technique, the coarse DQ can be measured by the dissimilarity between the original dataset  $D$  and the sanitized one  $D'$  by measuring, for example, in the case of transactional datasets, the difference between the item frequencies of the two datasets before and after the sanitization. Such dissimilarity can be estimated by the following expression:

$$Diss(D, D') = \frac{\sum_{i=1}^n |f_D(i) - f_{D'}(i)|}{\sum_{i=1}^n f_D(i)} \quad (7)$$

where  $i$  is a data item in the original database  $D$ , and  $f_D(i)$  is its frequency within the database, whereas  $i'$  is the given data item after the application of a privacy preservation technique and  $f_{D'}(i)$  is its new frequency within the transformed database  $D'$ . The same method can be used, extending the previous formula, also in the case of blocking techniques. If the data modification consists of aggregating some data values, the coarse DQ is given by the loss of detail in the data. As in the case of the  $k$ -Anonymity algorithm [15], given a database  $DB$  with  $N_A$  attributes and  $N$  transactions, if we identify a generalization scheme a domain generalization hierarchy  $GT$  with a depth  $h$ , it is possible to measure the coarse quality of a sanitized database  $SDB$  as:

$$Quality(SDB) = 1 - \frac{\sum_{i=1}^{i=N_A} \sum_{j=1}^{j=N} \frac{h}{|GT_{A_i}|}}{|DB| * |N_A|} \quad (8)$$

where  $\frac{h}{|GT_{A_i}|}$  represent the detail loss for each cell sanitized.

Once we have identified the *Best Set* we are able to apply our DQ-driven evaluation.

<sup>3</sup> <http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html>

We now present a three steps Evaluation Framework based on the previous concepts.

1. A set of “Interesting” PPDM’s is selected. These algorithms are tested on a generic database and evaluated according the general parameters (Efficiency, Scalability, Hiding failure, Coarse Data Quality, Level of privacy). The result of this step is a restricted set of *Candidate algorithms*
2. A test database with the same characteristics of the target database is generated. An IQM schema with the AIS and the related DMG is the result of this step.
3. The Information Driven DQ Evaluation Algorithm is applied in order to identify the algorithm that finally will be applied.

As it is probably obvious to the readers, the most “time consuming” step in terms of required user interactions is step 2. The design a good IQM is the core of our evaluation framework. We believe that a top down approach is, in this cases, the most appropriate. More in detail, the first task should be the identification of the high level information that is relevant and for which we are interested in measuring the impact of PPDM algorithms. It could also can be useful to involve in this task some authorized users (e.g. in case of Health DBA’s, doctors, etc.) in order to understand all the possible uses of the database and the relevance of the retrieved information. The use of datamining tools could be useful to identify non-evident aggregate information.

A second task would then, given the high level information, determine the different constraints (both simple and complex) and evaluate their relevance. Also in this case, discussions with authorized users and DB designers, and the use of DM tools (e.g. discover association rules) could help to build a good IQM. Finally it is necessary, by taking into account all the previous information, to rate the relevance of the attributes involved. This top down analysis is useful not only for the specific case of PPDM evaluation, but, if well developed, is a powerful tool to understand the real information contents, its value and the relation between the information stored in a given database. In the context of an “Information Society” this is a non negligible added value.

## 6 Conclusion

The Data Quality of Privacy Preserving Data Mining Algorithms is an open problem. In this paper we have proposed an approach to represent three important aspects of the data quality, an algorithm to magnify the impact of a PPDM algorithm on the data quality of a given database and a framework to select the most suitable algorithm for such database. We have also carried out extensive tests for the case of association rules PPDM algorithms. We plan to extend our work with some tests on other types of PPDM algorithms. Another direction that we plan to explore is the use and then the test of our framework over non-homogeneous algorithms. Because our principal focus is to provide tools supporting the database administrator in identifying the most suitable algorithm



for their database, we also plan to develop a tool that will be integrated with GADAET [16] in order to allow an intuitive Asset, AIS and DMG design and an automated algorithms test.

## References

1. V. S. Verykios, E. Bertino, I. Nai Fovino, L. Parasiliti, Y. Saygin, Y. Theodoridis. *State-of-the-art in Privacy Preserving Data Mining*. SIGMOD Record, 33(1):50-57 (2004).
2. S. R. M. Oliveira and O. R. Zaiane. *Privacy Preserving Frequent Itemset Mining*. In Proceedings of the 28th International Conference on Very Large Databases (2003).
3. S. R. M. Oliveira and O. R. Zaiane. *Privacy Preserving Clustering by Data Transformation*. In Proceedings of the 18th Brazilian Symposium on Databases, Manaus, Amazonas, Brazil, October 2003, pp.304-318.
4. University of Milan - Computer Technology Institute - Sabanci University. *COD-MINE*. IST project. 2002-2003.
5. D. Agrawal and C. C. Aggarwal. *On the Design and Quantification of Privacy Preserving Data Mining Algorithms*. In Proceedings of the 20th ACM Symposium on Principle of Database System (2001), pp. 247-255.
6. R. Agrawal and R. Srikant. *Privacy Preserving Data Mining*. In Proceedings of the ACM SIGMOD Conference of Management of Data (2000), pp. 439-450.
7. A. Evfimievski. *Randomization in Privacy Preserving Data Mining*. SIGKDD Explor. Newsl. (2002), Vol.4, n. 2, pp. 43-48, ACM
8. S. J. Rizvi and J. R. Haritsa. *Maintaining Data Privacy in Association Rule Mining*. In Proceedings of the 28th International Conference on Very Large Databases (2003).
9. G. Kumar Tayi, D. P. Ballou. *Examining Data Quality*. Comm. of the ACM, Vol. 41, Issue 2, pp. 54-58, Feb. 1998.
10. K. Orr. *Data Quality and System Theory*. Comm. of the ACM, Vol. 41, Issue 2, pp. 66-71, Feb. 1998.
11. Y. Wand and R. Y. Wang. *Anchoring Data Quality Dimensions in Ontological Foundations*. Comm. of the ACM, Vol. 39, Issue 11, pp. 86-95 Nov. 1996.
12. W. Kent. *Data and reality*. North Holland, New York, 1978.
13. A.V. Levitin and T.C. Redman. *Data as resource: properties, implications and prescriptions*. Sloan Management review, Cambridge, Fall 1998, Vol. 40, Issue 1, pp. 89-101.
14. R. Y. Wang, D. M. Strong. *Beyond Accuracy: what Data Quality Means to Data Consumers*. Journal of Manahement Information Systems Vol. 12, Issue 4, pp. 5-34, (1996).
15. L. Sweeney. *Achieving k-Anonymity Privacy Protection using Generalization and Suppression*. Journal on Uncertainty, Fuzzyness and Knowledge-based Sys., Vol. 10, Issue 5, pp. 571-588,(2002).
16. E. Bertino, I. Nai Fovino, L. Parasiliti Provenza. *A Framework for Evaluating Privacy Preserving Data Mining Algorithms*. Data Mining and Knowledge Discovery Journal, 2005.

# Essential Patterns: A Perfect Cover of Frequent Patterns

Alain Casali, Rosine Cicchetti, and Lotfi Lakhal

Laboratoire d'Informatique Fondamentale de Marseille (LIF),  
CNRS UMR 6166, Université de la Méditerranée  
Case 901, 163 Avenue de Luminy, 13288 Marseille Cedex 9, France  
`lastname@lif.univ-mrs.fr`

**Abstract.** The extraction of frequent patterns often yields extremely voluminous results which are difficult to handle. Computing a concise representation or cover of the frequent pattern set is thus an interesting alternative investigated by various approaches. The work presented in this article fits in such a trend. We introduce the concept of essential pattern and propose a new cover based on this concept. Such a cover makes it possible to decide whether a pattern is frequent or not, to compute its frequency and, in contrast with related work, to infer its disjunction and negation frequencies. A levelwise algorithm with a pruning step which uses the maximal frequent patterns for computing the essential patterns is proposed. Experiments show that when the number of frequent patterns is very high (strongly correlated data), the defined cover is significantly more reduced than the cover considered until now as minimal: the frequent closed patterns.

## 1 Introduction and Motivations

It is well known that frequent patterns mined from transactional databases can be extremely voluminous, and specially when data is strongly correlated. In such a context, it is difficult for the end-user to handle the extracted knowledge. Various approaches have addressed this problem and attempt to compute a concise representation, also called cover, of the whole set of frequent patterns [PBTL99, BR01, Pha02, CG02, BR03, KRG04]. Such a cover has a twofold interest: determining, at lower cost, (i) if an unknown pattern is frequent or not, and (ii) if it is the case, what is its frequency.

Unfortunately, among the covers proposed in the literature [KRG04], most of them are not proved to be concise representations and, in some cases, they can be more voluminous than the whole set of frequent patterns. In such cases, the initial objectives are not met and the difficulty to manage patterns worsens.

We call “*perfect cower*” of a set of frequent patterns a cover which is always smaller than this set. Only two have been proposed in the literature: the cover using frequent closed patterns [PBTL99, PHM00, Pha02, ZH02] and the one based on non derivable frequent patterns [CG02].

In this paper, we propose a new perfect cover based on the inclusion-exclusion identities [Nar82]. With this intention, we introduce the concept of essential pattern. We show that our representation based on frequent essential patterns is a perfect cover and has an interesting advantage when compared with the two other approaches: it is possible, not only, to retrieve the frequency of an unknown pattern but also to know the frequency of its disjunction and of its negation. Moreover, we propose a levelwise algorithm for computing the set of essential patterns. Through various experiments, we compare our approach with the one known as minimal: the cover based on closed patterns. Results are convincing since in the worse cases, when data is highly correlated, the size of our representation is significantly more reduced than the size of the closed pattern cover.

The remainder of the article is the following: in section 2 we recall the principle of the inclusion-exclusion identities. We use these identities in order to define a novel concept in section 3: the essential patterns. In the section 4, we propose the new cover based on the essential patterns. We propose a levelwise algorithm with a pruning step which uses the maximal frequent patterns for computing the frequent essential patterns in section 5. Experimental results are given in section 6. In conclusion, we resume the strengths of our contribution and the prospects for research.

## 2 Frequency Measures and Inclusion-Exclusion Identities

Let  $\mathcal{D}$  be a transactional database over a set of items and  $X \in \mathcal{P}(\mathcal{I})^1$  a pattern, we define three weight measures, which are compatible with the weight functions defined in [STB<sup>+</sup>02], for  $X$ : (i) its frequency (denoted by  $Freq(X)$ ), (ii) its disjunctive frequency (denoted by  $Freq(\vee X)$ ) and (iii) its negative frequency (denoted by  $Freq(\neg X)$ ). The disjunctive frequency of a pattern  $X$  can be seen as the probability to have at least one 1-pattern of  $X$  and the frequency of the negation stands for the probability to have no 1-pattern of  $X$ .

$$Freq(X) = \frac{|\{X' \in \mathcal{D} \mid X \subseteq X'\}|}{|\mathcal{D}|} \tag{1}$$

$$Freq(\vee X) = \frac{|\{X' \in \mathcal{D} \mid X \cap X' \neq \emptyset\}|}{|\mathcal{D}|} \tag{2}$$

$$Freq(\neg X) = \frac{|\{X' \in \mathcal{D} \mid X \cap X' = \emptyset\}|}{|\mathcal{D}|} \tag{3}$$

*Example 1.* - Let  $\mathcal{D}$  be the following database:

We have:  $Freq(AC) = 2/4$ ,  $Freq(\vee AC) = 1$  and  $Freq(\neg AC) = 0$ . Since  $Freq(\vee AC) = 1$ , each transaction in  $\mathcal{D}$  contains either the 1-pattern  $A$ , or the 1-pattern  $C$ , or both of them.

---

<sup>1</sup>  $\mathcal{P}(X)$  is the powerset of  $X$ .

**Table 1.** Database example  $\mathcal{D}$

$Tid$	Items
1	ABCD
2	ABD
3	CE
4	ACD

The inclusion-exclusion identities make it possible to state, for a pattern  $X$ , the relationship between the frequency, the frequency of the disjunction and the frequency of the negation, as follows:

$$Freq(X) = \sum_{\substack{X' \subseteq X \\ X' \neq \emptyset}} (-1)^{(|X'| - 1)} Freq(\vee X'). \tag{4}$$

$$Freq(\vee X) = \sum_{\substack{X' \subseteq X \\ X' \neq \emptyset}} (-1)^{(|X'| - 1)} Freq(X'). \tag{5}$$

$$Freq(\neg X) = 1 - Freq(\vee X) \text{ (from De Morgan Law)} \tag{6}$$

*Example 2.* - In our database example, we have:

1.  $Freq(AC) = Freq(A) + Freq(C) - Freq(\vee AC) = 3/4 + 3/4 - 1 = 2/4$
2.  $Freq(\vee AC) = Freq(A) + Freq(C) - Freq(AC) = 3/4 + 3/4 - 2/4 = 1$
3.  $Freq(\neg AC) = 1 - Freq(\vee AC) = 0$ .

Computing the frequency of the disjunction for a pattern can be performed along with computing its frequency and thus the execution time of levelwise algorithms is not altered. Provided with the frequency of the disjunction for the frequent patterns, a perfect cover of frequent patterns can be defined and the computation of the negation frequency is straightforward (*cf.* De Morgan Law).

### 3 Essential Patterns

A pattern  $X$  is essential if and only if its disjunctive frequency is different from the disjunctive frequency of all its direct subsets. Since the disjunctive frequency function is an increasing monotone function, we do not need to examine the disjunctive frequency of each direct subset for a pattern  $X$ . Checking that the disjunctive frequency of  $X$  is different to the greatest disjunctive frequency of its direct subsets is a sufficient condition to be sure that  $X$  is an essential pattern. A more formal definition of the concept of essential pattern is given below. Then, we show that the constraint “ $X$  is an essential pattern” is an antimonotone constraint for the inclusion. Thus, this constraint is compatible with the frequency constraint and makes it possible to use levelwise algorithms for mining frequent essential patterns.

**Definition 1. (Essential patterns)** - Let  $\mathcal{D}$  be a transactional database over a set of items  $\mathcal{I}$  and let  $X \in \mathcal{P}(\mathcal{I})$  be a pattern. We say that  $X \neq \emptyset$  is an essential pattern if and only if

$$Freq(\vee X) \neq \max_{x \in X} (Freq(\vee X \setminus x)). \quad (7)$$

Let us denote by  $\mathcal{E}$  the set of essential patterns and  $\mathcal{E}(\mathbb{F})$  the set of frequent essential patterns<sup>2</sup>.

*Example 3.* - In our database example, the pattern  $AC$  is an essential pattern because  $Freq(\vee AC) \neq Freq(\vee A)$  and  $Freq(\vee AC) \neq Freq(\vee C)$ .

**Lemma 1.** - Let us consider the two following constraints: “ $X$  is frequent” ( $C_1$ ) and “ $X$  is essential” ( $C_2$ ). The conjunction of the two constraints is antimonotone for the inclusion (i.e. if  $X$  is a frequent essential pattern, then all its subsets are frequent and essential patterns).

## 4 Frequency Computation Using an Improvement of the Inclusion-Exclusion Identities

The three following formulas show firstly how to compute the frequency of the disjunction from the set of essential patterns and secondly how to optimize the inclusion-exclusion identities for finding efficiently the frequency of a frequent pattern. A naive method for computing the frequency of a pattern  $X$  requires the knowledge of the disjunctive frequency of all its subsets. Formula 8 shows how we can derive the disjunctive frequency of any patterns using only essential patterns.

**Lemma 2.** Let  $X$  be a set of items, then we have:

$$Freq(\vee X) = \max_{Y \in \mathcal{E}} \{Freq(\vee Y) \mid Y \subseteq X\}. \quad (8)$$

The formula 9 is an optimization based on the concept of essential patterns and the formula 10 is an original method for the derivation of the frequency of  $X$ .

**Lemma 3.** -  $\forall X \in \mathcal{P}(\mathcal{I})$ , let be  $Y \in \text{Argmax}(\{Freq(\vee X') \mid X' \subseteq X \text{ and } X' \in \mathcal{E}\})$ , then we have:

$$Freq(X) = \sum_{\substack{X' \subseteq X \\ X' \neq \emptyset}} (-1)^{|X'|-1} \begin{cases} Freq(\vee Y) & \text{if } Y \subseteq X' \\ Freq(\vee X') & \text{elsewhere} \end{cases} \quad (9)$$

**Theorem 1.** -  $\forall X \in \mathbb{F}$ ,  $X \notin \mathcal{E}(\mathbb{F})$ , let be  $Y \in \text{Argmax}(\{Freq(\vee X') \mid X' \subseteq X \text{ and } X' \in \mathcal{E}(\mathbb{F})\})$ , then we have:

<sup>2</sup>  $\mathbb{F}$  is the set of frequent patterns.

$$Freq(X) = \sum_{\substack{X' \subseteq X \\ X' \neq \emptyset \\ X' \not\subseteq Y}} (-1)^{|X'|-1} Freq(\vee X') \tag{10}$$

The set of essential patterns is not sufficient to define a perfect cover for the set of frequent patterns because we cannot decide if an unknown pattern is frequent or not. That is why we add the positive border for the frequency constraint to the set of frequent essential patterns for testing if an unknown pattern is frequent or not. If it is frequent, then theorem 1 makes it possible to compute the frequency of its conjunction. Thus, the set of frequent essential patterns  $(\mathcal{E}(\mathbb{F}))$  increased with the positive border for the frequency constraint  $(BD^+(\mathbb{F}))$  is a perfect cover for the frequent patterns.

**Definition 2. (Perfect cover)** - Let  $\mathcal{D}$  a transactional database over a set of items  $\mathcal{I}$  (each transaction is a subset of  $\mathcal{I}$ ) and  $\mathbb{F}$  the set of frequent patterns. We say that  $\mathbb{G}$  is a cover for  $\mathbb{F}$  if and only if the frequency of each element of  $\mathbb{F}$  can be retrieved by using only patterns of  $\mathbb{G}$  ( $\forall X \in \mathbb{F}, \mathbb{G} \models Freq(X)$ ). Moreover, if  $\mathbb{G} \subseteq \mathbb{F}$ , the cover is called perfect.

**Theorem 2.** - Let  $BD^+(\mathbb{F})$  be the positive border (i.e. the set of maximal frequent patterns) and  $\mathcal{E}(\mathbb{F})$  the set of essential frequent patterns, then  $BD^+(\mathbb{F}) \cup \mathcal{E}(\mathbb{F})$  is a perfect cover for the frequent patterns.

## 5 The MEP Algorithm

For finding the frequent essential patterns, we propose a levelwise algorithm with a pruning step which uses the maximal frequent patterns  $(BD^+(\mathbb{F}))$ . The algorithm MEP (Mining Essential Patterns) includes the function `Max_Set_Algorithm` which discovers maximal frequent patterns (e.g. Max-Miner [Bay98], Gen-Max [GZ01]).

*Example 4.* The perfect cover of our example for the threshold “ $Minfreq = 2/4$ ” is the following: the set of frequent essential patterns is given in table 2 and the positive border  $BD^+(\mathbb{F})$  is given in table 3.

We know that the pattern  $ABD$  is frequent because it belongs to the positive border. Let us compute its frequency.

**Table 2.** Frequent essential pattern for “ $Freq(X) \geq 2/4$ ”

Essential pattern	Disjunctive frequency
$A$	$3/4$
$B$	$2/4$
$C$	$3/4$
$D$	$3/4$
$AC$	$1$
$CD$	$1$

**Table 3.** Positive border for “ $Freq(X) \geq 2/4$ ”

Positive border
ABD
ACD

---

**Algorithm 1** MEP Algorithm

---

```

1:  $BD^+(\mathbb{F}) := \text{Max\_Set\_Algorithm}(\mathcal{D}, \text{Minfreq})$ 
2:  $L_1 = \{\text{frequent 1-pattern}\}$ 
3:  $i := 1$ 
4: while  $L_i \neq \emptyset$  do
5:    $C_{i+1} := \text{Gen\_Apriori}(L_i)$ 
6:    $C_{i+1} := \{X \in C_{i+1} \mid \exists Y \in BD^+(\mathbb{F}) : X \subseteq Y\}$ 
7:   Scan the database for mining the disjunctive frequency  $\forall X \in C_{i+1}$ 
8:    $L_{i+1} := \{X \in C_{i+1} \mid \nexists x \in X : Freq(\vee X) = Freq(\vee X \setminus x)\}$ 
9:    $i := i + 1$ 
10: end while
11: return  $\bigcup_{j=1..i} L_j$ 

```

---

- We use lemma 2 to find its disjunctive frequency:  $Freq(\vee ABD) = \max(Freq(\vee A), Freq(\vee B), Freq(\vee D)) = Freq(\vee A) = Freq(\vee D) = 3/4$ .
- We apply theorem 1 to compute its frequency:

The patterns  $A$  and  $D$  are two frequent essential patterns included in  $ABD$  for which the disjunctive frequency is maximal. Thus we have:  $Argmax(\{Freq(\vee ABD) \mid X' \subseteq X \text{ and } X' \in \mathcal{E}(\mathbb{F})\}) = \{A, D\}$ . We need one of these two patterns to apply theorem 1, we choose  $Y = A$ . We obtain the following equality:  $Freq(ABD) = Freq(A) + Freq(B) + Freq(D) - Freq(\vee AB) - Freq(\vee AD) - Freq(\vee BD) + Freq(\vee ABD) = Freq(B) + Freq(D) - Freq(\vee BD)$ . Since the pattern  $BD$  is not an essential, we need to know its disjunctive frequency. By applying, once more, theorem 1, we obtain  $Freq(B) + Freq(D) - Freq(\vee BD) = Freq(B)$ . Accordingly,  $Freq(ABD) = 2/4$ .

We have eliminated all the patterns included between  $A$  and  $ABD$  in the inclusion-exclusion identities because the sum of their disjunctive frequencies, weighted of the good coefficient, is null.

## 6 Experimental Results

By providing the disjunctive and the negative frequencies, the proposed approach enriches the results obtained with the two other perfect covers proposed in the literature. Our objective is now to show, through various experiments, that the size of this new cover is often smaller than the size of the cover based on the

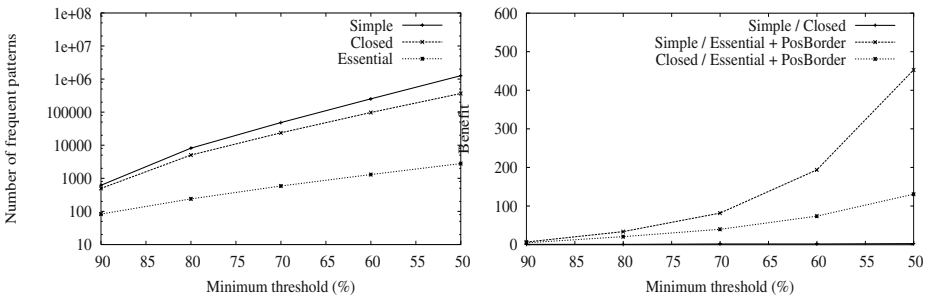
**Table 4.** Datasets

Name	Number of transactions	Average size of each transaction	Number of items
CHES	3 196	37	75
CONNECT	65 557	43	129
PUMSB	49 046	74	2 113
PUMSB*	49 046	50,5	2 088

frequent closed patterns and this in the most critical cases: strongly correlated data. For meeting this objective, we evaluate the number of frequent essential patterns and compare it with the number of frequent closed patterns by using four datasets<sup>3</sup>. The characteristics of the datasets used for experiments are given in table 4. They are:

- the dataset CHES,
- the dataset CONNECT,
- the datasets of census PUMSB and PUMSB\*, extracted from « PUMS sample file ». PUMSB\* is the same dataset than PUMSB from which are removed all the patterns which have a threshold greater or equal to 80%,

For all the experiments, we choose relevant minimum thresholds. In these four datasets, only encompassing strongly correlated data, the ratio between frequent patterns and the total number of patterns is high. Thus we are in the most difficult cases. For finding the positive border we use Gen-Max algorithm [GZ01]. In the dataset PUMSB\*, using either the frequent closed patterns or the frequent essential patterns as a cover is advantageous: the gain compared to the set of frequent patterns for the dataset PUMSB\* with the threshold *Minfreq* = 20% is about 45. On the other hand, for this dataset, even if the approach by essential patterns is better than the one with closed patterns, the obtained gain is near to one. In the three remaining datasets, the approach by essential is very efficient. With the dataset CHES, many of frequent patterns are closed patterns, but the number of essential patterns is relatively small. This results in a benefit, for the



**Fig. 1.** Experimental results for CHES

<sup>3</sup> <http://fimi.cs.helsinki.fi/>



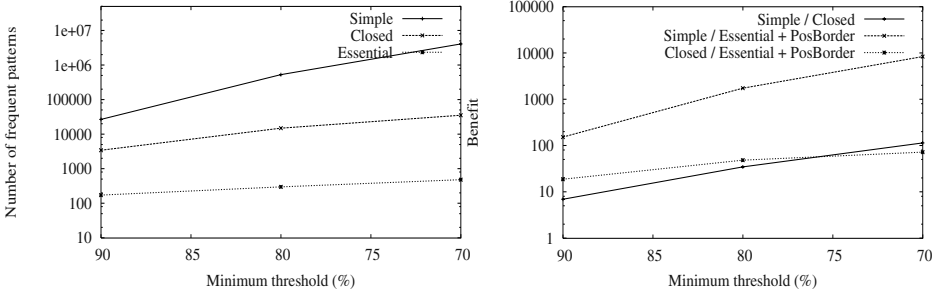


Fig. 2. Experimental results for CONNECT

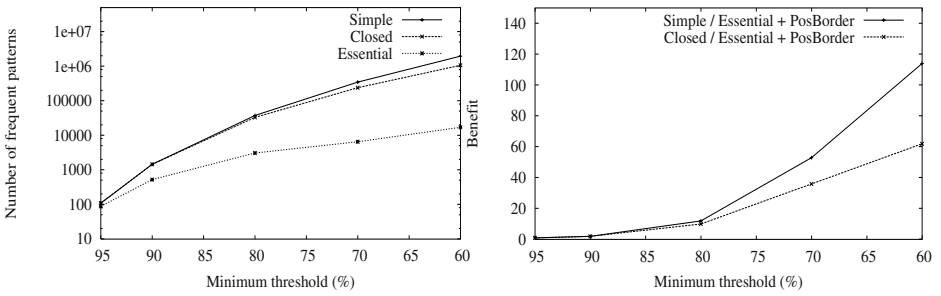


Fig. 3. Experimental results for PUMBS

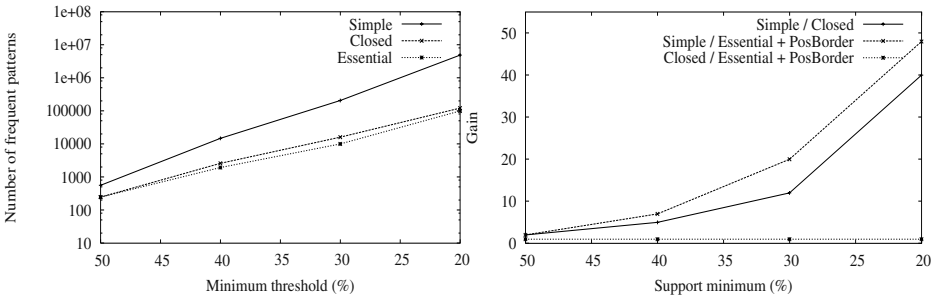


Fig. 4. Experimental results for PUMBS\*

threshold  $Minfreq = 50\%$ , of a factor 40 compared to the original approach and of a factor 20 compared to the approach using closed patterns. With the dataset CONNECT and a threshold  $Minfreq = 70\%$ , the benefit compared to frequent patterns is approximately of a factor 2500 and compared to closed frequent pattern of a factor 20. We can see that with the dataset PUMSB, the benefit compared to the approach by frequent closed patterns is of a factor 20 for a threshold  $Minfreq = 60\%$  and compared to the approach by frequent patterns is approximately 40.

For more readability in the figures, we have omitted “frequent patterns” in the legends. Thus “simple” means frequent patterns, “closed” stands for frequent closed pattern and “essential” symbolizes frequent essential pattern.

## 7 Conclusion

In this paper, we propose a novel perfect cover for the frequent patterns based on the inclusion-exclusion identities. We introduce the concept of essential pattern. The perfect cover is based (i), on one hand, on the positive border which can be used to determine if an unknown pattern is frequent or not, and (ii), on the other hand, on the frequent essential patterns which make it possible to derive the frequency of a frequent pattern by applying an optimization of the inclusion-exclusion identities. Compared with the existing perfect covers, our method makes it possible to mine at lower cost, along with the frequency of a frequent pattern, the frequency of its disjunction and negation. We have also shown, from an experimental point of view, the efficiency of our approach in the most critical cases: when the mined data is correlated, the set of frequent patterns is extremely voluminous. Having a perfect cover is specially interesting to quickly answer the ad hoc requests of decision makers.

Concerning future work, it would be interesting to define the disjunctive closure operator to reduce the number of essential patterns because this concept is similar to the concept of key. Thus, by applying this operator to each frequent essential pattern, we would also obtain a set of disjunctive closed patterns which could be a perfect cover for the frequent patterns. Since closure operators are surjective functions, the number of frequent disjunctive closed patterns will be thus lower than the number of frequent essential patterns.

## Acknowledgement

We would like to thank Stéphane Lopes who has performed the experiments.

## References

- [Bay98] Roberto Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the International Conference on Management of Data, SIGMOD*, pages 85–93, 1998.
- [BR01] Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *Proceedings of the 20th Symposium on Principles of Database Systems, PODS*, pages 267–273, 2001.
- [BR03] Artur Bykowski and Christophe Rigotti. Dbc: a condensed representation of frequent patterns for efficient mining. In *Information Systems*, volume 28(8), pages 949–977, 2003.
- [CG02] Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In *In Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, PKDD*, pages 74–85, 2002.

- [GZ01] Karam Gouda and Mohammed Javeed Zaki. Efficiently Mining Maximal Frequent Itemsets. In *Proceedings of the 1st IEEE International Conference on Data Mining, ICDM*, pages 3163–170, 2001.
- [KRG04] Marzena Kryszkiewicz, Henryk Rybinski, and Marcin Gajek. Dataless transitions between concise representations of frequent patterns. In *Journal of Intelligent Information System*, volume 22(1), pages 41–70, 2004.
- [Nar82] Hiroshi Narushima. Principle of Inclusion-Exclusion on Partially Order Sets. In *Discrete Mathematics*, volume 42, pages 243–250, 1982.
- [PBT99] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory, ICDT*, pages 398–416, 1999.
- [Pha02] Viet Phan Luong. The closed keys base of frequent itemsets. In *Proceedings of the 4th Data Warehousing and Knowledge Discovery, DAWAK*, pages 181–190, 2002.
- [PHM00] Jian Pei, Jiawei Han, and Runying Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In *Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD*, pages 21–30, 2000.
- [STB<sup>+</sup>02] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing Iceberg Concept Lattices with Titanic. In *Data and Knowledge Engineering*, volume 42(2), pages 189–222, 2002.
- [ZH02] Mohammed Javeed Zaki and Ching-Jui Hsio. CHARM: An Efficient Algorithm for Closed Itemset Mining. In *Proceedings of the 2nd SIAM International Conference on Data mining*, 2002.

# Processing Sequential Patterns in Relational Databases

Xuequn Shang<sup>1</sup> and Kai-Uwe Sattler<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Magdeburg,  
P.O.BOX 4120, 39106 Magdeburg, Germany  
shang@iti.cs.uni-magdeburg.de

<sup>2</sup> Department of Computer Science and Automation, Technical University of Ilmenau  
kus@tu-ilmenau.de

**Abstract.** Database integration of data mining has gained popularity and its significance is well recognized. However, the performance of SQL based data mining is known to fall behind specialized implementation since the prohibitive nature of the cost associated with extracting knowledge, as well as the lack of suitable declarative query language support. Recent studies have found that for association rule mining and sequential pattern mining with carefully tuned SQL formulations it is possible to achieve performance comparable to systems that cache the data in files outside the DBMS. However most of the previous pattern mining methods follow the method of *Apriori* which still encounters problems when a sequential database is large and/or when sequential patterns to be mined are numerous and long.

In this paper, we present a novel SQL based approach that we recently proposed, called *Prospad* (PROjection Sequential PAttern Discovery). *Prospad* fundamentally differs from an *Apriori*-like candidate set generation-and-test approach. This approach is a pattern growth-based approach without candidate generation. It grows longer patterns from shorter ones by successively projecting the sequential table into subsequential tables. Since a projected table for a sequential pattern  $i$  contains all and only necessary information for mining the sequential patterns that can grow from  $i$ , the size of the projected table usually reduces quickly as mining proceeds to longer patterns. Moreover, avoiding creating and dropping cost of some temporary tables, depth first approach is used to facilitate the projecting process.

## 1 Introduction

An important data mining issue is the discovery of sequential patterns, which involves finding frequent subsequences as patterns in a sequence database. Most of the algorithms used today typically employ sophisticated in-memory data structures, where the data is stored into and retrieved from flat files. While the mined datasets are often stored in relational format, the integration of data mining with relational database systems is an emergent trend in database research

and development area [4]. There are several potential advantages of a SQL implementation. One can make use of the powerful mechanisms for accessing, filtering, and indexing data, as well as SQL parallelization the database systems provided. In addition, SQL-aware data mining systems have ability to support ad-hoc mining, ie. allowing to mine arbitrary query results from multiple abstract layers of database systems or Data Warehouses.

However, from the performance perspective, data mining algorithms that are implemented with the help of SQL are usually considered inferior to algorithms that process data outside the database systems. One of the important reasons is that offline algorithms employ sophisticated in-memory data structures and try to reduce the scan of data as few times as possible while SQL-based algorithms either require several scans over the data or require many and complex joins between the input tables. This fact motivated us to develop a new SQL-based algorithm which avoids making multiple passes over the large original input table and complex joins between the tables.

The remainder of this paper is organized as follows. In section 2, we briefly discuss sequential pattern mining algorithms and implementations employing SQL queries. *Prospad* algorithm is explained in section 3. Section 4 presents several experiments that assess the performance of the algorithms based on synthetic datasets. Related works is illustrated in section 5. We conclude the paper in section 6 and give a brief outlook on future work.

## 2 Sequential Pattern Mining with SQL

### 2.1 Problem Statement

The sequential pattern mining problem can be formally defined as follows. Let  $I = \{i_1, i_2, \dots, i_m\}$  be a set of items. An itemset is a subset of items. A sequence  $s = (s_1, s_2, \dots, s_n)$  is an ordered list of itemsets, where  $s_i \subseteq I$ ,  $i \in \{1, \dots, m\}$ . The number of itemsets in a sequence is called the length of the sequence. The length  $l$  of a sequence  $s = (s_1, s_2, \dots, s_n)$  is defined as follows.

$$l = \sum_{i=1}^m |s_i|$$

A sequence  $s_a = (a_1, a_2, \dots, a_n)$  is a subsequence of another sequence  $s_b = (b_1, b_2, \dots, b_m)$  if there exist integers  $1 \leq i_1 < i_2 < \dots < i_n \leq m$  such that  $a_1 \subseteq b_{i_1}$ ,  $a_2 \subseteq b_{i_2}$ , ...,  $a_n \subseteq b_{i_n}$ .

A sequence database  $D$  is a set of tuples  $(cid, tid, itemset)$ , where  $cid$  is a customer id,  $tid$  is a transaction id based on the transaction time. The support of a sequence  $s$  in a sequence database  $D$ , denoted as  $sup_D(s)$ , is the number of tuples in the database containing  $s$ . Given a support threshold  $min\_supp$ , a sequence  $s$  is called a frequent sequential pattern in  $D$  if  $sup_D(s) \geq min\_supp$ . Given a sequence database and the  $min\_supp$ , the problem of mining sequential patterns is to discover all frequent sequential patterns in the database.

## 2.2 Algorithms for Mining Sequential Patterns

There are several algorithms for mining sequential patterns. These algorithms can be classified into two categories: *Apriori-based* [1,9,3] and *Pattern-growth* [6,2] methods.

- *Apriori-based* approaches are based on an anti-monotone *Apriori* heuristic: if any length  $k$  pattern is not frequent in the database, its super-pattern of length  $(k + 1)$  can never be frequent. They begin with the discovery of frequent 1-sequences, and then generate the set of potential frequent  $(k + 1)$ -sequences from the set of frequent  $k$ -sequences. This kind of algorithm, though reducing search space, may still suffer from the following three nontrivial, inherent costs:
  - It is costly to handle a huge number of candidate sets.
  - It is tedious to repeatedly scan the database.
  - It generates a combinatorially explosive number of candidates when mining large sequential patterns.
- *Pattern-growth* methods are a more recent approach to deal with the problems of mining sequential patterns. The key idea is to avoid repeatedly scanning the entire database and testing and generating large set of candidates, and to focus the search on a restricted portion of the initial database. *Prefixspan* [6] is the most promising of the *Pattern-growth* approaches. It recursively projects a sequence database into a set of smaller projected sequence databases and mines locally frequent patterns in each projected database. *Prefixspan* achieves high efficiency, compared with *Apriori-based* approaches.

## 2.3 Sequential Pattern Mining Based on SQL

Almost all previous sequential pattern mining algorithms with SQL are based on *Apriori*, which consist of a sequence of steps proceeding in a bottom-up manner. The result of the  $k$ th step is the set of frequent itemsets, denoted as  $F_k$ . The first step computes frequent 1-itemsets  $F_1$ . The candidate generation phase computes a set of potential frequent  $k$ -itemsets  $C_k$  from  $F_{k-1}$ . The support counting phase filters out those itemsets from  $C_k$  that appear more frequently in the given set of transactions than the minimum support and stores them in  $F_k$ .

Before data can be mined with SQL, it has to be made available as relational tables. Sequence data, as the input, is transformed into the first normal form table  $T$  with three column attributes: sequence identifier (*cid*), transaction identifier (*tid*) and item identifier (*item*). For a given *cid*, typically there are multiple rows in the sequence table corresponding to different items that belong to transactions in the data sequence. The output is a collection of frequent sequences. The schema of the frequent sequences table is  $(item_1, eno_1, \dots, item_k, eno_k, len)$ . The *len* attribute gives the length of the sequence. The *eno* attributes stores the element number of the corresponding items.

```

insert into  $C_k$  select  $I_1.item_1, I_1.eno_1, \dots, I_1.item_{k-1}, I_1.eno_{k-1}, I_2.item_{k-1}, I_1.eno_{k-1} +$ 
 $I_2.eno_{k-1} - I_2.eno_{k-2}$ 
from  $F_{k-1}I_1, F_{k-1}I_2$ 
where  $I_1.item_2 = I_2.item_1$  and  $\dots$  and
 $I_1.item_{k-1} = I_2.item_{k-2}$  and
 $I_1.eno_3 - I_1.eno_2 = I_2.eno_2 - I_2.eno_1$  and  $\dots$  and
 $I_1.eno_{k-1} - I_1.eno_{k-2} = I_2.eno_{k-2} - I_2.eno_{k-3}$ 

```

**Fig. 1.** Candidate generation phase in SQL-92

In [10], Thomas et al. addressed the problem of mining sequential patterns using SQL queries and developed SQL formulations. The statement for generating  $C_k$  from  $F_{k-1}$  in SQL-92, is shown in Figure 1.

A well known approach for support counting using SQL-92 presented in [10], is similar with K-Way joins for association rules [7]. However, [10] adds additional constraints including *window-size*, *max-gap*, and *min-gap* besides the constraint of *min\_supp*.

[10] points out that it is possible to express complex sequential pattern mining computations in SQL. The approach, however, shares similar strengths and weaknesses as the *Apriori* method. For frequent pattern mining, a SQL-based frequent pattern growth method called *Propad* [8] has been developed for efficient mining frequent pattern in relational database systems. The general idea of *Propad* is to successively transform the original transaction table into a set of frequent item-related projected tables, then to separately mine each one of the tables as soon as they are built. In this paper, we explore the spirit of *Propad* for mining sequence patterns.

### 3 Prospad: PROjection Sequential Pattern Discovery in SQL

In this section, a novel SQL-based sequential pattern mining method, called *Prospad*, which recursively generates a set of frequent sequence-related projected tables, then mine locally frequent patterns in each projected table, is illustrated using an example.

Let us give an example with four sequences in Figure 2(a). Sequence data, as the input, is transformed into the first normal form table  $T$  with three column attributes: sequence identifier (*cid*), transaction identifier (*tid*) and item identifier (*item*) as shown in Figure 2(b).

Before the new algorithm is given, let us give some definitions as follows.

**Definition 1.** Given a sequence pattern  $p$  and a frequent item  $i$  in the sequence database  $D$ , a sequence-extended pattern can be formed by adding the item  $i$  to its prefix sequence  $p$ , and an itemset-extended pattern can be formed by adding the item  $i$  to the last element of the prefix sequence  $p$ .

(a) <i>A Sequence Database</i>	(b) $\tau$																																																																																																																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">CID</th> <th style="text-align: center;">TID</th> <th style="text-align: center;">Sequence</th> </tr> </thead> <tbody> <tr><td style="text-align: center;">1</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>a</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>a, b, c</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>a, c</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>d</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">5</td><td style="text-align: center;"><i>c, f</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>a, d</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>b, c</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>a, e</i></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>e, f</i></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>a, b</i></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>d, f</i></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">5</td><td style="text-align: center;"><i>b</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>e</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>g</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>a, f</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;"><i>b</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">6</td><td style="text-align: center;"><i>c</i></td></tr> </tbody> </table>	CID	TID	Sequence	1	1	<i>a</i>	1	2	<i>a, b, c</i>	1	3	<i>a, c</i>	1	4	<i>d</i>	1	5	<i>c, f</i>	2	1	<i>a, d</i>	2	2	<i>c</i>	2	3	<i>b, c</i>	2	4	<i>a, e</i>	3	1	<i>e, f</i>	3	2	<i>a, b</i>	3	3	<i>d, f</i>	3	4	<i>c</i>	3	5	<i>b</i>	4	1	<i>e</i>	4	2	<i>g</i>	4	3	<i>a, f</i>	4	4	<i>c</i>	4	5	<i>b</i>	4	6	<i>c</i>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">CID</th> <th style="text-align: center;">TID</th> <th style="text-align: center;">Item</th> </tr> </thead> <tbody> <tr><td style="text-align: center;">1</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>a</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>a</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>b</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>a</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>d</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">5</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">1</td><td style="text-align: center;">5</td><td style="text-align: center;"><i>f</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>a</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">1</td><td style="text-align: center;"><i>d</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>b</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>a</i></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>e</i></td></tr> <tr><td style="text-align: center;">...</td><td style="text-align: center;">...</td><td style="text-align: center;">...</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">4</td><td style="text-align: center;"><i>c</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">5</td><td style="text-align: center;"><i>b</i></td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">6</td><td style="text-align: center;"><i>c</i></td></tr> </tbody> </table>	CID	TID	Item	1	1	<i>a</i>	1	2	<i>a</i>	1	2	<i>b</i>	1	2	<i>c</i>	1	3	<i>a</i>	1	3	<i>c</i>	1	4	<i>d</i>	1	5	<i>c</i>	1	5	<i>f</i>	2	1	<i>a</i>	2	1	<i>d</i>	2	2	<i>c</i>	2	3	<i>b</i>	2	3	<i>c</i>	2	4	<i>a</i>	2	4	<i>e</i>	...	...	...	4	4	<i>c</i>	4	5	<i>b</i>	4	6	<i>c</i>
CID	TID	Sequence																																																																																																																													
1	1	<i>a</i>																																																																																																																													
1	2	<i>a, b, c</i>																																																																																																																													
1	3	<i>a, c</i>																																																																																																																													
1	4	<i>d</i>																																																																																																																													
1	5	<i>c, f</i>																																																																																																																													
2	1	<i>a, d</i>																																																																																																																													
2	2	<i>c</i>																																																																																																																													
2	3	<i>b, c</i>																																																																																																																													
2	4	<i>a, e</i>																																																																																																																													
3	1	<i>e, f</i>																																																																																																																													
3	2	<i>a, b</i>																																																																																																																													
3	3	<i>d, f</i>																																																																																																																													
3	4	<i>c</i>																																																																																																																													
3	5	<i>b</i>																																																																																																																													
4	1	<i>e</i>																																																																																																																													
4	2	<i>g</i>																																																																																																																													
4	3	<i>a, f</i>																																																																																																																													
4	4	<i>c</i>																																																																																																																													
4	5	<i>b</i>																																																																																																																													
4	6	<i>c</i>																																																																																																																													
CID	TID	Item																																																																																																																													
1	1	<i>a</i>																																																																																																																													
1	2	<i>a</i>																																																																																																																													
1	2	<i>b</i>																																																																																																																													
1	2	<i>c</i>																																																																																																																													
1	3	<i>a</i>																																																																																																																													
1	3	<i>c</i>																																																																																																																													
1	4	<i>d</i>																																																																																																																													
1	5	<i>c</i>																																																																																																																													
1	5	<i>f</i>																																																																																																																													
2	1	<i>a</i>																																																																																																																													
2	1	<i>d</i>																																																																																																																													
2	2	<i>c</i>																																																																																																																													
2	3	<i>b</i>																																																																																																																													
2	3	<i>c</i>																																																																																																																													
2	4	<i>a</i>																																																																																																																													
2	4	<i>e</i>																																																																																																																													
...	...	...																																																																																																																													
4	4	<i>c</i>																																																																																																																													
4	5	<i>b</i>																																																																																																																													
4	6	<i>c</i>																																																																																																																													

**Fig. 2.** A sequence database and its relational format

**Table 1.** An example projected table

CID	TID	Item
1	3	<i>a</i>
1	3	<i>c</i>
1	4	<i>d</i>
1	5	<i>c</i>
1	5	<i>f</i>
2	3	<i>b</i>
2	3	<i>c</i>
2	4	<i>a</i>
2	4	<i>e</i>
3	5	<i>b</i>
4	5	<i>b</i>
4	6	<i>c</i>

For example, if we have a sequence pattern  $\{a, c\}$ , and a frequent item  $b$ , then  $\{a, c, b\}$  is a sequence-extended pattern and  $\{a, (c, b)\}$  is an itemset-extended pattern.

**Definition 2.** Given a sequence table  $T$ , a frequent sequence  $s$ -related projected table, denoted as  $PT_s$ , has three column attributes: sequence identifier ( $cid$ ),



transaction identifier (*tid*), item identifier (*item*), which collects all sequences containing *s*. Moreover, in the sequence containing *s*, only the subsequence prefixed with the first occurrence of *a* should be considered.

For sequence *c* in *T*, its projected table  $PT_c$  is shown in Table 1.

In order to avoid repetitiousness and to ensure each frequent item is projected to at most one projected table, we suppose items in alphabetical order. The mining process can be regarded as a process of frequent sequence growth, which is facilitated by projecting sequence tables in a top-down fashion. The whole process is as follows:

- At the first level we simply gather the count of each item and items that satisfy the minimum support are inserted into the transformed transaction table *TF* that has the same schema as transaction table *T*. It means that only frequent-1 items are included in the table *TF*. The SQL statements used to create the table *TF* are illustrated as follows. We use **select distinct** before the **group by** to ensure that only distinct data-sequences are counted.

```
insert into F select s.item, count(*)
from      (select distinct item, cid from T) as s
group by  item
having    count(*) ≥ min_supp

insert into TF select T.cid, T.tid, T.item
from      T, F
where     T.item = F.item
```

- At the second level, for each frequent-1 item *i* in the table *TF* we construct its respective projected table  $PT_i$ . This is done by two phases. The first step finds all sequences in the *TF* containing *i*, in which only the subsequence prefixed with the first occurrence of *i* should be collected. This can be expressed in SQL as follows. We use a temporary table *TEMP\_id* to collect the first occurrence of *i* in each sequence containing *i*.

```
insert into TEMP_id select cid, min(tid) as tid
from      TF
where     item = i
group by  cid

insert into PT_i (select t1.*
from      TF t1, TEMP_id t2
where     t1.cid = t2.cid and
          t1.tid > t2.tid

union
select * from TF
where     (cid, tid) in (select cid, tid from TEMP_id) and
          item > i)
```

The second step finds all items that could be an itemset-extended pattern in the  $PT_i$ . All the items that occur in the same transaction as  $i$  can itemset-extend the current pattern. And then we update all these items by appending '-' to distinct an itemset-extended pattern from a sequence-extended pattern. The SQL statements can be expressed as follow:

```

insert into TEMP_item select cid, tid, min(item) as item
from      PT_i
where     (cid, tid) in (select cid, tid from TEMP_id)
group by  cid, tid

update PT_i
set      item = item || '-'
where   (cid, tid, item) in (select cid, tid, item from TEMP_item)

```

Then find local frequent items. Frequent-1 items are regarded as the prefixes, frequent-2 patterns are gained by simply combining the prefixes and their local frequent itemsets. For instance, we get the frequent-1 items  $\{a, b, c, d, e, f\}$ , their respective projected tables  $PT_a, PT_b, PT_c, PT_d, PT_e, PT_f$ . For the table  $PT_a$ , its local frequent item are  $\{a, b, b-, c, d, f\}$ , frequent-2 patterns are  $\{\{a, a\}, \{a, b\}, \{(a, b)\}, \{a, c\}, \{a, d\}, \{a, f\}\}$ .

- At the next level, for each frequent item  $j$  in the projected table  $PT_i$  we recursively construct its projected table  $PT_{i,j}$  and gain its local frequent items. A projected table is filtered if there is no frequencies can be derived. For instance, no local frequent itmes in the  $PT_{a,a}$  can be found, the processing for mining frequent sequential patterns associated with  $aa$  terminates.

Basically, the projecting process can be facilitated either by breadth first approach or by depth first approach. In breadth first approach, each frequent item has its corresponding projected table and local frequent itemsets table at level  $k$ . That is,  $n$  projected tables need to be generated if we have  $n$  frequent itemsets at level  $k$ . It is obviously unpracticable because too many temporary tables have to be held especially for dense database and for low support threshold.

Avoiding creating and dropping cost of some temporary tables, we use depth first approach. In fact, the temporary tables such as  $TEMP\_id$ ,  $TEMP\_item$  and frequent itemsets table  $F$  are only required in the constructing projection tables  $PT$ . So they can be cleared for constructing the other  $PTs$  after one  $PT$  is constructed. Moreover, in the whole mining procedure, the  $PT$  tables of each frequent item are not constructed together. The mining process for each frequent item is independent of that for others. In that case, we only need one  $PT$  table at the each level. The number of  $PT$  tables is the same magnitude as the length of maximum frequent pattern.

## 4 Performance Evaluation

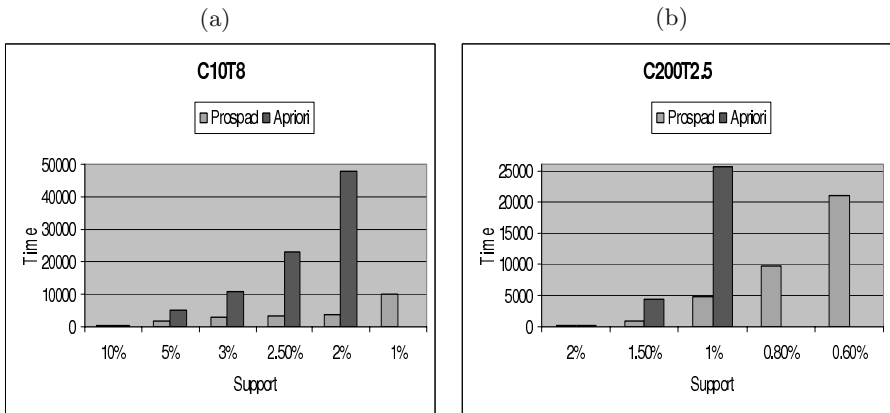
To evaluate the efficiency of our algorithm *Prospad*, we have done extensive experiments on various kinds of datasets with different features by comparing with *A priori*-based approach in SQL.

### 4.1 Datasets

We use synthetic sequence data generation with program described in Apriori algorithm paper [1] for experiments. The nomenclature of these data sets is of the form CwwTxxSyyIzz, where ww denotes the total number of sequences in K (1000's). xx denotes the average number of items in a transaction and yy denotes the average number of transactions in a sequence. On average, a frequent sequential pattern consists of four transactions, and each transaction is composed of zz items. We report experimental results on two data sets, they are respectively C10T8S8I8 and C200T2.5S10I1.25.

### 4.2 Performance Comparison

In this subsection, we describe our algorithm performance compared with K-Way joins. Our experiment were performed on DBMS IBM DB2 EEE V8. The performance measure was the execution time of the algorithm on the datasets with different support threshold.



**Fig. 3.** Comparison of two approaches. In (a), for K-Way join approach with the support value of 1%, in (b), with the support value of 0.8%, the running times were so large that we had to abort the runs in many cases.

Figure 3 (a) through (b) show the total time taken by the two approaches on the two data sets respectively: *K-way* joins, *Prospad*. From the graph we can make the following observations: as the support is high, the frequent sequences are short and the number of sequence is not large. The advantages of *Prospad* over *Apriori* are not so impressive. *Prospad* is even slightly worse than *Apriori*. For example, the maximal length of frequent patterns is 1 and the number of frequent sequences is 89 when the datasets is C10T8S8I8 with the support threshold 10%, *Apriori* can finish the computation shorter than the time for *Prospad*. However, as the support threshold goes down, the gap

is becoming wider: *Prospad* is much faster than *Apriori*. When the support is low, the number as well as the length of frequent sequences increase dramatically. Candidate sequences that *Apriori* must handle becomes extremely large, joining the candidate sequences with sequence tables becomes very expensive. In contrast, *Prospad* avoids candidates generation and test. That's why *Prospad* can get significant performance improvement.

## 5 Related Works

The sequential pattern mining problem was first introduced by Agrawal and Srikant in [1]. Recently researchers have started to focus on issues to integrating mining with databases. There have been language proposals to extend SQL to support mining operators. The Data Mining Query Language DMQL [5] proposed a collection of such operators for classification rules, characteristics rule, association rules, etc. In [11], Wojciechowski proposed an SQL-like language capable of expressing queries concerning all classes of patterns.

There are some SQL-based approaches proposed to mine sequential patterns in [10], for example *K-Way Joins*, Subquery-based. These new algorithms are on the base of *Apriori*-like approach. They use the same statement for generating candidate k-itemsets and differ only in the statements used for support counting. [10] also use object-relational extensions in SQL like UDFs, BLOBs, table function etc. to improve performance.

## 6 Summary and Conclusion

In this paper, we propose an efficient SQL based algorithm to mine sequential patterns from relational database systems. Rather than *Apriori*-like method it adopts the divide-and-conquer strategy and projects the sequence table into a set of frequent item-related projected tables. Experimental study shows that the *Prospad* algorithm can get higher performance than *K-Way joins* based on *Apriori*-like especially on large and dense data sets.

There remain lots of further investigations. We plan to do more experimentation on different datasets, including real datasets, to consolidate the experiences in mining all classes of patterns with SQL. We try to explore *Prospad* for mining closed and maximal sequential patterns.

## References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.
2. C. Antunes and A. L. Oliveira. Sequential pattern mining algorithms: Trade-offs between speed and memory. In *Second International Workshop on Mining Graphs, Trees and Sequences*, Pisa, Italy, September 2004.

3. J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435, New York, NY, USA, 2002. ACM Press.
4. S. Chaudhuri. Data mining and database systems: Where is the intersection? *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21(1), March 1998.
5. J. Han, Y. Fu, and W. Wang. Dmql: A data mining query language for relational database. In *Proc. Of the 1996 SIGMOD workshop on research issues on data mining and knowledge discovery*, Montreal, Canada, 1996.
6. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns efficiently by prefixprojected pattern growth. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages =.
7. S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 345–354, Seattle, WA, June 1998. ACM Press.
8. X. Shang and K. Sattler. Depth-first frequent itemset mining in relational databases. In *Proc. ACM Symposium on Applied Computing SAC 2005*, New Mexico, USA, 2005.
9. R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.
10. S. Thomas and S. Sarawagi. Mining generalized association rules and sequential patterns using SQL queries. In *Knowledge Discovery and Data Mining*, pages 344–348, 1998.
11. M. Wojciechowski. Mining various patterns in sequential data in an sql-like manner. In *ADBIS (Short Papers)*, pages 131–138, 1999.

# Optimizing a Sequence of Frequent Pattern Queries\*

Mikołaj Morzy, Marek Wojciechowski, and Maciej Zakrzewicz

Poznań University of Technology,  
Institute of Computing Science,  
ul. Piotrowo 3a, 60-965 Poznań, Poland  
{mmorzy, marek, mzakrz}@cs.put.poznan.pl

**Abstract.** Discovery of frequent patterns is a very important data mining problem with numerous applications. Frequent pattern mining is often regarded as advanced querying where a user specifies the source dataset and pattern constraints using a given constraint model. A significant amount of research on efficient processing of frequent pattern queries has been done in recent years, focusing mainly on constraint handling and reusing results of previous queries. In this paper we tackle the problem of optimizing a sequence of frequent pattern queries, submitted to the system as a batch. Our solutions are based on previously proposed techniques of reusing results of previous queries, and exploit the fact that knowing a sequence of queries a priori gives the system a chance to schedule and/or adjust the queries so that they can use results of queries executed earlier. We begin with simple query scheduling and then consider other transformations of the original batch of queries.

## 1 Introduction

Discovery of frequent patterns is a very important data mining problem with numerous practical applications. The two most prominent classes of patterns are frequent itemsets [1] and sequential patterns [3]. Informally, frequent itemsets are subsets frequently occurring in a collection of sets of items, and sequential patterns are the most frequently occurring subsequences in sequences of sets of items.

Frequent pattern mining is often regarded as advanced querying where a user specifies the source dataset, the minimum frequency threshold (called *support*), and optionally pattern constraints within a given constraint model [7]. A significant amount of research on efficient processing of frequent pattern queries has been done in recent years, focusing mainly on constraint handling and reusing results of previous queries in the context of frequent itemsets and sequential patterns.

In this paper we tackle the problem of optimizing a sequence of frequent pattern queries, submitted to the system at the same time or within a short time window. Our approach is motivated by data mining environments working in batch mode, where users submit batches of queries to be scheduled for execution. However, the

---

\* This work was partially supported by the grant no. 4T11C01923 from the State Committee for Scientific Research (KBN), Poland.

techniques discussed in the paper can also be applied to systems following the on-line interactive mining paradigm if we allow the system to group the queries received within a given time window and process them as batches. Our solutions are based on previously proposed techniques of reusing results of previous frequent pattern queries, and exploit the fact that knowing a sequence of queries a priori gives the system a chance to schedule and/or adjust the queries so that they can use results of queries executed earlier. We begin with simple query scheduling and then discuss other possibilities of transforming the original batch of queries.

The paper is organized as follows. In Section 2 we review related work. Section 3 contains basic definitions regarding frequent pattern queries and relationships between them. In Section 4 we present our new technique of optimizing batches of frequent pattern queries. Section 5 contains conclusions and directions for future work.

## 2 Related Work

Multiple-query optimization has been extensively studied in the context of database systems (e.g., [13]). The idea was to identify common subexpressions and construct a global execution plan minimizing the overall processing time by executing the common subexpressions only once for the set of queries. Data mining queries could also benefit from this general strategy, however due to their different nature they require novel multiple-query processing methods.

Within the data mining context, multiple-query optimization has not drawn much attention so far. As an introduction to multiple data mining query optimization, we can regard techniques of reusing intermediate or final results of previous queries to answer a new query. Methods falling into that category are: incremental mining, caching intermediate query results, and reusing materialized results of previous queries provided that syntactic differences between the queries satisfy certain conditions.

Incremental mining was first studied in the context of frequent itemsets in [5], where the *FUP* algorithm was proposed. Incremental mining consist in efficiently discovering frequent patterns in an incremented dataset, exploiting previously discovered frequent patterns. After the pioneer *FUP* algorithm, several other incremental mining algorithms were proposed for itemsets and sequential patterns.

An interesting solution based upon the idea of reusing intermediate results of previous queries was proposed in [10] by introducing the concept of a knowledge cache that would keep recently discovered frequent itemsets along with their support value, in order to facilitate interactive and iterative mining. In [8], the authors postulated to cache only non-redundant itemsets like closed itemsets [11].

Syntactic differences between data mining queries, representing situations when one query can be efficiently answered using the results of another query, have been first analyzed in [4] for association rule queries. The authors identified three relationships between the queries: equivalence, inclusion, and dominance, and provided appropriate query processing algorithms exploiting the relationships.

In [9], we proposed to materialize results of frequent pattern queries rather than rule queries, motivated by the fact that generation of rules from patterns is a straightforward and relatively inexpensive task. The results of previous queries were stored in the form of materialized data mining views. Syntactic differences between frequent pattern queries considered in the paper included one leading to the possibility of incremental mining, and one analogous to the inclusion relationship from [4]. Those syntactic differences and the corresponding relationships between pattern queries were later more thoroughly analyzed by us in [16] within an example constraint model for frequent itemset discovery.

To the best of our knowledge, the only two real multiple-query processing methods for frequent patterns are *Apriori Common Counting (ACC)* and *MineMerge*, proposed by us in [14] and [15] respectively. Unfortunately, both methods have significant drawbacks that limit their practical applications. *ACC* is bound to the *Apriori* algorithm [2], which is a serious limitation since *Apriori* has been outperformed by newer, pattern-growth algorithms (see [6] for an overview). Moreover, *ACC* requires more memory than its base algorithm – *Apriori*. On the other hand, *MineMerge* is not bound to a particular mining algorithm but has been proven non-deterministic, sometimes resulting in longer processing time than in case of sequential query processing. Therefore, our goal in this paper is introduction of a new method, not bound to any mining methodology, having memory requirements not greater than applied base mining algorithm, and guaranteeing performance gains, at least under certain assumptions.

### 3 Frequent Pattern Queries

In this section, we provide a universal definition of a frequent pattern query, and review the possibilities of reusing frequent pattern queries' results. We generalize the definitions and methods introduced by us for frequent itemset mining [16], and reformulate them so that they can serve as a basis for modeling batches of frequent pattern queries and development of batch optimization (scheduling) techniques.

#### 3.1 Basic Definitions

**Definition 1 (Frequent pattern query and its predicates).** A frequent pattern query is a tuple  $fpq = (R, a, \Sigma, \Phi, \beta)$ , where  $R$  is a relation,  $a$  is an attribute of  $R$ ,  $\Sigma$  is a condition involving the attributes of  $R$  (called *database predicate*),  $\Phi$  is a condition involving discovered patterns (called *pattern predicate*),  $\beta$  is the minimum support threshold. The result of the  $fpq$  is a set of patterns discovered in  $\pi_a \sigma_{\Sigma} R$ , satisfying  $\Phi$ , and having support  $\geq \beta$ .

**Example 1.** Given the database relation  $R_I(a_1, a_2)$ , where  $a_2$  is a set-valued attribute and  $a_1$  is of integer type. The frequent pattern query  $fpq_1 = (R_I, "a_2", "a_1 > 5", "itemset1 < 4", 3\%)$  describes the problem of discovering frequent itemsets in the



set-valued attribute  $a_2$  of the relation  $R_1$ . The frequent itemsets with support of at least 3% and length less than 4 are discovered in the collection of records having  $a_1 > 5$ .

### 3.2 Relationships Between Frequent Pattern Queries

**Definition 2 (Identical queries).** Two frequent pattern queries  $fpq_1$  and  $fpq_2$  are *identical* if they both operate on the same attribute  $a$  of the same relation  $R$  and  $\Sigma_1 = \Sigma_2$  and  $\Phi_1 = \Phi_2$  and  $\beta_1 = \beta_2$ .

We assume the existence of some canonical form for database predicates and pattern predicates, to which all query predicates will be transformed before any optimization takes place. Thus, we can assume that two queries will guarantee to return the same results for any database instance only if the queries are identical.

**Definition 3 (Data set inclusion).** Given two frequent pattern queries  $fpq_1=(R, a, \Sigma_1, \Phi_1, \beta_1)$  and  $fpq_2=(R, a, \Sigma_2, \Phi_2, \beta_2)$ . We say that  $fpq_1$  *includes data set* of  $fpq_2$  (denoted as  $fpq_2 \subseteq_d fpq_1$ ) if for each possible instance of the relation  $R$ ,  $\sigma_{\Sigma_2}R \subseteq \sigma_{\Sigma_1}R$ .

**Definition 4 (Pattern set inclusion).** Given two frequent pattern queries  $fpq_1=(R, a, \Sigma_1, \Phi_1, \beta_1)$  and  $fpq_2=(R, a, \Sigma_2, \Phi_2, \beta_2)$ . We say that  $fpq_1$  *includes pattern set* of  $fpq_2$  (denoted as  $fpq_2 \subseteq_p fpq_1$ ) if for each possible instance of the relation  $R$ , all the patterns returned by  $fpq_2$  will also be returned by  $fpq_1$ , and for each pattern returned by both queries, its counted support value will be the same for both queries.

In terms of query predicates,  $fpq_1$  includes pattern set of  $fpq_2$  if  $\beta_1 \leq \beta_2$  and  $\Phi_1$  is a relaxation of  $\Phi_2$  (denoted as  $\Phi_2 \subseteq \Phi_1$ ). (Formal definition of such relaxation will depend on a particular pattern constraint model. The only requirement is that the relaxation relationship should be a partial order on pattern sets. See [16] for an example constraint model.)

It should be noted that the above relationships between frequent pattern queries are defined in terms of query predicates, independently of the contents of the mined database. Therefore, we can assume that for a given particular constraint model and a given frequent pattern query language, the system will be able to discover the relationships between the queries within a batch, just by analyzing the syntactic differences between the queries. Although, for a flexible constraint model and/or query language, the task might not be trivial, it is considered an implementation issue and as such is beyond the scope of this paper.

**Example 2.** Given the database relation  $R_1(a_1, a_2)$  from Example 1 and two frequent pattern queries:  $fpq_1 = (R_1, "a_2", "a_1 > 5", "itemset \leq 4", 4\%)$  and  $fpq_2 = (R_1, "a_2", "a_1 < 3", "itemset \leq 5", 2\%)$ , we have  $fpq_1 \subseteq_p fpq_2$ , and no data set inclusion relationship between the queries.

### 3.3 Reusing Results of Previous Frequent Pattern Queries

According to the analysis from [16] a frequent pattern query  $fpq_2$  can be efficiently answered using known (materialized) results of another query  $fpq_1$  provided that  $fpq_1 \subseteq_d fpq_2$  (i.e.,  $fpq_2$  operates on an incremented data set) and  $fpq_2 \subseteq_p fpq_1$  (i.e., the

pattern selection condition and the minimum support threshold of  $fpq_2$  are not more restrictive than those of  $fpq_1$ ).

The general algorithm for answering  $fpq_2$  using the results of  $fpq_1$ , where  $fpq_1 \subseteq_d fpq_2$  and  $fpq_2 \subseteq_p fpq_1$ , consists of two steps:

- 1) Result Filtering (RF) by removing the patterns returned by  $fpq_1$  that do not satisfy the pattern selection condition and the minimum support threshold of  $fpq_2$ .
- 2) Incremental Mining (IM) using pattern selection condition and the minimum support threshold of  $fpq_2$ , treating the data set of  $fpq_2$  as incremented data set of  $fpq_1$  for which the patterns of  $fpq_2$ 's interest are known from the previous step.

It should be noted that there are three particular cases, in which one of or even both the above steps can be omitted:

- If  $\beta_1 = \beta_2$  and  $\Phi_1 = \Phi_2$  then the filtering step is not needed;
- If  $\Sigma_1 = \Sigma_2$  then the incremental mining step is not needed;
- If  $\Sigma_1 = \Sigma_2$  and  $\beta_1 = \beta_2$  and  $\Phi_1 = \Phi_2$  then the results of  $fpq_2$  are equal to the results of  $fpq_1$  and therefore neither filtering nor incremental mining is needed.

Regarding the implementation details and costs of the two steps of the above query result reusing algorithm, we observe that the first step (RF) is a simple scan of the query results, inexpensive both in terms of computations (simple conditions on patterns) and I/O (query results are typically several orders of magnitude smaller than the queries' source dataset). As for the second step (IM), it is obvious that different incremental mining techniques can be (or have already been) developed for various types of patterns, constraint models, and mining methodologies. However, as a reference incremental pattern mining method, we can regard the partition-based incremental mining technique described in [16], exploiting the well-known ideas of partition-based mining [12].

The partition-based incremental frequent pattern mining technique logically divides the database into two partitions: (1) the records covered by the query  $fpq_1$  ( $\sigma_{\Sigma_1}R$ ) – for this partition the locally frequent patterns are known, (2) the records covered by the query  $fpq_2$ , and not covered by  $fpq_1$  ( $\sigma_{\Sigma_2}R - \sigma_{\Sigma_1}R$ ). The method begins with discovering patterns locally frequent in the second partition. Next, based on the property of partition-based mining, locally frequent patterns from both partitions are used as candidate patterns for the whole  $fpq_2$ 's data set ( $\sigma_{\Sigma_2}R$ ), and counted in one scan of the data set.

In typical scenarios, incremental mining is more efficient than running a complete mining algorithm, and result filtering is significantly more efficient than incremental mining. Therefore, if for a given query there are results of a previous query that can be reused, the system should reuse them rather than run a complete mining algorithm. If more than one query's results are applicable, the system should first look for the possibility of Result Filtering (on the smallest available pattern set), and then, if RF is not possible, the system should opt for Incremental Mining or the combination of RF and IM involving the smallest increment of the dataset.

## 4 Optimizing Batches of Frequent Pattern Queries

The problem of optimizing batches of frequent pattern queries can be informally defined as follows: Given a batch (a sequence) of frequent pattern queries, find the execution plan that minimizes the total execution time of the whole batch. In this paper, we consider optimization techniques based on the idea of reusing some query's results to answer other queries. Within this framework, we develop a novel multiple-query optimization method for batches of frequent pattern queries starting with simple query scheduling, and then considering other transformations of the original batch. We assume that the batches of queries to be optimized contain no duplicates. Elimination of duplicates should be one of the pre-processing steps, right after the transformation of queries' predicates to the canonical form used by the system, which is required to determine the relationships between the queries.

### 4.1 Query Scheduling

For the purpose of modeling batches of frequent pattern queries, let us start with a formal definition of the relationship between queries capturing the possibility of reusing other queries' results:

**Definition 5 (Result reusing).** Given two frequent pattern queries:  $fpq_1=(R, a, \Sigma_1, \Phi_1, \beta_1)$  and  $fpq_2=(R, a, \Sigma_2, \Phi_2, \beta_2)$ . We say that  $fpq_2$  can reuse results of  $fpq_1$  (denoted as  $fpq_1 \rightarrow fpq_2$ ) if  $\sigma_{\Sigma_1}R \subseteq \sigma_{\Sigma_2}R$  and  $\Phi_2 \subseteq \Phi_1$  and  $\beta_1 \leq \beta_2$ .

**Theorem 1.** The relationship of result reusing is a partial order on a set (batch) of frequent pattern queries.

*Proof.* The relationship of result reusing is defined upon three partial order relationships on query predicates. To prove that the relationship of result reusing is also a partial order, we have to prove that it is reflexive, anti-symmetric, and transitive (from the definition of partial order). The three properties of the relationship of result reusing can be derived from its definition and inherent properties of the partial orders upon which it is built in the following way:

- $\sigma_{\Sigma_1}R \subseteq \sigma_{\Sigma_1}R \wedge \Phi_1 \subseteq \Phi_1 \wedge \beta_1 \leq \beta_1 \Rightarrow fpq_1 \rightarrow fpq_1$  (proof of reflexivity);
- $(fpq_1 \rightarrow fpq_2 \wedge fpq_2 \rightarrow fpq_1) \Rightarrow (\sigma_{\Sigma_1}R \subseteq \sigma_{\Sigma_2}R \wedge \Phi_2 \subseteq \Phi_1 \wedge \beta_1 \leq \beta_2 \wedge \sigma_{\Sigma_2}R \subseteq \sigma_{\Sigma_1}R \wedge \Phi_1 \subseteq \Phi_2 \wedge \beta_2 \leq \beta_1) \Rightarrow (\sigma_{\Sigma_1}R = \sigma_{\Sigma_2}R \wedge \Phi_2 = \Phi_1 \wedge \beta_1 = \beta_2) \Rightarrow (\Sigma_1 = \Sigma_2 \wedge \Phi_2 = \Phi_1 \wedge \beta_1 = \beta_2) \Rightarrow (fpq_1 = fpq_2)$  (proof of anti-symmetry);
- $(fpq_1 \rightarrow fpq_2 \wedge fpq_2 \rightarrow fpq_3) \Rightarrow (\sigma_{\Sigma_1}R \subseteq \sigma_{\Sigma_2}R \wedge \Phi_2 \subseteq \Phi_1 \wedge \beta_1 \leq \beta_2 \wedge \sigma_{\Sigma_2}R \subseteq \sigma_{\Sigma_3}R \wedge \Phi_3 \subseteq \Phi_2 \wedge \beta_2 \leq \beta_3) \Rightarrow (\sigma_{\Sigma_1}R \subseteq \sigma_{\Sigma_3}R \wedge \Phi_3 \subseteq \Phi_1 \wedge \beta_1 \leq \beta_3) \Rightarrow (fpq_1 \rightarrow fpq_3)$  (proof of transitivity).

Based on the above result reusing relationship, we propose the initial multiple-query optimization method for pattern queries, consisting in scheduling the batch of queries according to the result reusing relationship.

**Algorithm 1 (Multiple-Query Optimization Using Query Scheduling)**

**Input:** a set of pattern queries  $FPQ = \{fpq_1, fpq_2, \dots, fpq_n\}$  searching for frequent patterns in the  $a$  attribute of the database relation  $R$

**Output:** results of queries from  $FPQ$

1. sort  $FPQ$  according to the result reusing relationship to form a schedule  $SFPQ = (sfpq_1, sfpq_2, \dots, sfpq_n)$  where for each  $fpq_i \in FPQ$  there exist  $sfpq_j \in SFPQ$  such that  $fpq_i = sfpq_j$  and for each pair  $sfpq_i, sfpq_j$  of queries in  $SFPQ$ :  $sfpq_i \rightarrow sfpq_j \Rightarrow i < j$ ;
2. **for**  $i := 1$  **to**  $n$  **do**
3.      $MPQ = \{sfpq_k : sfpq_k \rightarrow sfpq_i\}$ ;
4.     **if**  $MPQ = \emptyset$  **then**
5.         execute  $sfpq_i$  using a complete mining algorithm;
6.     **else**
7.         select  $mpq \in MPQ$  for which the estimated cost of reusing its results to answer  $sfpq_i$  is minimal; /\* see Section 3.3 \*/
8.         execute  $sfpq_i$  reusing the results of  $mpq$ ; /\* RF + IM, RF, or IM \*/
9.     **end if**;
10. **end for**;

**Rationale:** Sorting the queries according to the result reusing relationship guarantees that for each query  $sfpq_i$  all the queries whose results  $sfpq_i$  can reuse will be executed earlier. Thus, the algorithm maximizes the chances of efficiently answering the queries using available results of previous queries. (Note that since the result reusing relationship is a partial order, a topological sort algorithm has to be used, and in general more than one optimal schedule is possible.)

**4.2 Query Scheduling with Addition of Intermediate Queries**

Algorithm 1 can be regarded as an initial solution that optimizes processing of the batch of queries by introducing a query scheduling step. To identify further optimization possibilities, let us model a batch of pattern queries as a directed graph, in which the nodes represent queries and the edges represent the possibility on reusing the results of one query by another query.

**Definition 6 (Query Reusing Graph).** A directed graph  $QRG = (V, E)$  is a *query reusing graph* for the set of frequent pattern queries  $FPQ$  if and only if  $V = FPQ$ ,  $E = \{(fpq_i, fpq_j) \mid fpq_i, fpq_j \in FPQ \wedge fpq_i \rightarrow fpq_j \wedge (\nexists fpq_k \in FPQ \text{ such that } fpq_i \rightarrow fpq_k \wedge fpq_k \rightarrow fpq_j)\}$ .

Let us consider a database relation  $R_1(a, b)$  and an example batch of frequent pattern queries  $FPQ_1 = \{fpq_1, fpq_2, fpq_3, fpq_4, fpq_5, fpq_6\}$ , where  $fpq_1 = (R_1, a, "10 < b < 20", "true", 1\%)$ ,  $fpq_2 = (R_1, a, "10 < b < 30", "length(pattern) < 3", 2\%)$ ,  $fpq_3 = (R_1, a, "10 < b < 30", "true", 5\%)$ ,  $fpq_4 = (R_1, a, "10 < b < 30", "length(pattern) < 4", 4\%)$ ,  $fpq_5 = (R_1, a, "10 < b < 30", "true", 3\%)$ ,  $fpq_6 = (R_1, a, "0 < b < 20", "true", 1\%)$ . Figure 1 shows the query reusing graph for the batch of frequent pattern queries  $FPQ_1$ . To support the analysis of possible optimizations, edges of the graph have been labeled with corresponding query reusing methods.

Let us look at the queries  $fpq_2$  and  $fpq_5$  which can be answered using the results of  $fpq_1$  in two steps: RF (using a different support threshold for each of the two queries) and IM (with exactly the same increment of the data set for the two queries). For a single query, if both RF and IM are required, it is more beneficial to start with RF and then run IM with the more restrictive pattern predicate and support threshold. However, if we know that more than one query will require the IM task on the same incremented data set as one of its execution steps, then typically it should be better to start with the IM step using the pattern predicate and support threshold that will allow all the involved queries to reuse the results of that IM step using RF procedures.

Identified common IM tasks can be represented as appropriate intermediate queries added to the original batch. Obviously, in this case the system will have to answer more queries than requested by users but as long as the total number of IM steps for the batch is reduced, the overall execution time of the batch should be shortened. (Recall that RF is typically by several orders of magnitude more efficient than IM.)

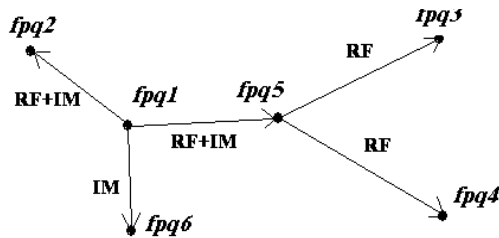


Fig. 1. Sample query reusing graph

For the example batch  $FPQ_I$  whose query reusing graph is presented in Fig. 1, we can provide the opportunity for reducing the number of executed IM tasks by adding an extra query  $fpq_7=(R_1, a, "10 < b < 30", "true", 2\%)$ . Figure 2 presents the query reusing graph for the extended batch  $FPQ_I' = FPQ_I \cup \{fpq_7\}$ .

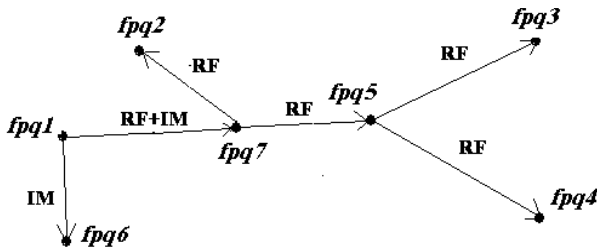


Fig. 2. Query reusing graph after the addition of the intermediate query

In general, such an intermediate query should have the same database constraint as the queries whose processing it going to improve, the support threshold equal to

the minimal support threshold among the queries, and pattern predicate being a logical alternative of the queries' pattern predicates. Based on the above observation, below we present an improved batch processing algorithm as an extension of Algorithm 1:

**Algorithm 2 (Multiple-Query Optimization Using Query Scheduling with Intermediate Queries)**

**Input:** a set of pattern queries  $FPQ = \{fpq_1, fpq_2, \dots, fpq_n\}$  searching for frequent patterns in the  $a$  attribute of the database relation  $R$

**Output:** results of queries from  $FPQ$

1. **for each**  $fpq_i \in FPQ$  **do**
2.  $IMQ_i = \{fpq_k : fpq_i \rightarrow fpq_k \wedge \Sigma_k \neq \Sigma_i \wedge \text{for all } fpq_x, fpq_y \in IMQ_i : \Sigma_x = \Sigma_y\};$
3. **if**  $|IMQ_i| > 1$  **then**
4.  $FPQ := FPQ \cup \{(R, a, \Sigma_{IMQ}, \Phi_{IMQ}, \beta_{IMQ})\}$ , where  $\Sigma_{IMQ}$  is the database predicate of queries from  $IMQ_i$ ,  $\Phi_{IMQ}$  is the logical alternative of pattern predicates of all queries from  $IMQ_i$ ,  $\beta_{IMQ}$  is the minimal support threshold among the queries from  $IMQ_i$ ;
5. **end if**;
6. **end for**;
7. execute Algorithm 1 for  $FPQ$

**Rationale:** An appropriate intermediate query is added for each set of queries that can reuse results of the same query using IM, provided that the set contains more than one query. As explained earlier, addition of each intermediate query to the batch reduces the number of IM tasks in the execution plan generated for the batch, which are typically much more costly than RF tasks.

### 4.3 Memory Management for Batch Execution

According to Algorithms 1 and 2, each of the pattern queries from a batch is executed using one of the three following methods: RF, IM, or complete mining. Taking into account that: (1) the most memory-consuming step of IM is execution of a base complete mining algorithm on the increment of the data set, and (2) RF can filter the patterns reading them from the disk one by one, we can say that memory requirements of our batch processing algorithms are not greater than in case of using a complete mining algorithm for all the queries in a batch, which is a desirable property.

Nevertheless, if possible within the memory limits, it will be beneficial for our technique to keep in main memory the results of queries that can be reused by some of the next queries (according to the generated schedule). As frequent pattern query results are typically much smaller than main memory structures used by pattern mining algorithms, such result caching introduces a negligible memory overhead. Moreover, once the system determines that the results of any of the previously executed queries cannot be reused by any queries to be executed later, the query's results can be removed from main memory, thus reducing the memory consumption.

## 5 Conclusions

In this paper we considered the problem of optimizing batches of frequent pattern queries. We presented a novel optimization technique based on techniques of reusing results of previous queries, previously proposed in literature. Our method exploits the fact that knowing a sequence of queries a priori gives the system a chance to schedule and/or adjust the batch of queries maximizing for each query the possibilities of reusing results of queries executed earlier.

The method proposed in this paper was motivated by data mining systems working in batch mode. In the future, we plan to focus on multiple-query optimization techniques oriented towards interactive systems, allowing dynamic addition of new queries to the set of currently optimized pattern queries.

## References

1. Agrawal R., Imielinski T., Swami A: Mining Association Rules Between Sets of Items in Large Databases. Proc. of the 1993 ACM SIGMOD Conf. on Management of Data (1993)
2. Agrawal R., Srikant R.: Fast Algorithms for Mining Association Rules. Proc. of the 20th Int'l Conf. on Very Large Data Bases (1994)
3. Agrawal R., Srikant R.: Mining Sequential Patterns. Proc. 11th ICDE Conf. (1995)
4. Baralis E., Psaila G.: Incremental Refinement of Mining Queries. Proceedings of the 1st DaWaK Conference (1999)
5. Cheung D.W., Han J., Ng V., Wong C.Y.: Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. Proc. of the 12th ICDE (1996)
6. Han J., Pei J.: Mining Frequent Patterns by Pattern-Growth: Methodology and Implications. SIGKDD Explorations, December 2000 (2000)
7. Imielinski T., Mannila H.: A Database Perspective on Knowledge Discovery. Communications of the ACM, Vol. 39, No. 11 (1996)
8. Jeudy B., Boulicaut J-F.: Using condensed representations for interactive association rule mining. Proceedings of the 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (2002)
9. Morzy T., Wojciechowski M., Zakrzewicz M.: Materialized Data Mining Views. Proceedings of the 4th PKDD Conference (2000)
10. Nag B., Deshpande P.M., DeWitt D.J.: Using a Knowledge Cache for Interactive Discovery of Association Rules. Proc. of the 5th KDD Conference (1999)
11. Pasquier N., Bastide Y., Taouil R., Lakhal L.: Discovering frequent closed itemsets for association rules. Proc. 7<sup>th</sup> Int'l Conf. On Database Theory (1999)
12. Savasere A., Omiecinski E., Navathe S.: An Efficient Algorithm for Mining Association Rules in Large Databases, Proc. 21th Int'l Conf. Very Large Data Bases (1995)
13. Sellis T.: Multiple-query optimization. ACM Transactions on Database Systems, Vol. 13, No. 1 (1988)
14. Wojciechowski M., Zakrzewicz M.: Evaluation of Common Counting Method for Concurrent Data Mining Queries. Proc. of the 7th ADBIS Conference (2003)
15. Wojciechowski M., Zakrzewicz M.: Evaluation of the Mine Merge Method for Data Mining Query Processing. Proc. of the 8th ADBIS Conference (2004)
16. Zakrzewicz M., Morzy M., Wojciechowski M.: A Study on Answering a Data Mining Query Using a Materialized View. Proceedings of the 19th ISCIS Conference (2004)

# A General Effective Framework for Monotony and Tough Constraint Based Sequential Pattern Mining

Enhong Chen<sup>1</sup>, Tongshu Li<sup>1</sup>, and Phillip C-y Sheu<sup>2</sup>

<sup>1</sup>Department of Computer Science and Technology,  
University of Science and Technology of China, Hefei, Anhui, 230027, P.R. China

<sup>2</sup>Department of EECS University of California,  
Irvine, CA 92697

cheneh@ustc.edu.cn, tsli@mail.ustc.edu.cn, psheu@uci.edu

**Abstract.** Sequential pattern mining has now become an important data mining problem. For many practical applications, the users may be only interested in those sequential patterns satisfying some constraints expressing their interest. The proposed constraints in general can be categorized into four classes, among which monotony and tough constraints are the most difficult ones to be processed. However, many of the available algorithms are proposed for some special constraints based sequential pattern mining. It is thus difficult to be adapted to other classes of constraints. In this paper we propose a new general framework called CBPSAlgm based on the projection-based pattern growth principal. Under this framework, ineffective item pruning strategies are designed and integrated to construct effective algorithms for monotony and tough constraint based sequential pattern mining. Experimental results show that our proposed methods outperform other algorithms.

## 1 Introduction

Recently, sequential pattern mining which discovers frequent subsequences as patterns in a sequence database has become an important data mining problem. There has been extensive research reported in this field. The proposed sequence pattern mining algorithms can be categorized into 3 classes. The first is Apriori-based horizontal formatting method, a typical algorithm is GSP [1]. The second is Apriori-based vertical formatting method, such as SPADE [2]. The third is projection-based pattern growth method, such as PrefixSpan [3].

For many practical applications, the users may be only interested in those sequential patterns satisfying some constraints expressing their interest. For example, in the analysis of telecom warning sequences, the users may just care about the warning sequential patterns at serious warning level and occurring in some special district. If all warning sequential patterns are mined first and filtered with these constraints then, the sequential pattern mining efficiency may be reduced greatly.

Up till now, many different types of constraints have been proposed, such as item constraint, length constraint, super-pattern constraint, aggregate constraint, regular expression constraint, duration constraint and gap constraint, and so on. These constraints in general can be categorized into four classes: anti-monotony, monotony,



succinctness and tough constraint [4]. Among these four constraints, succinctness constraint can be easily processed. Anti-monotony constraint has Apriori trick property. The pruning power of Apriori trick can be exploited completely. In fact all of the sequential pattern mining problems exist the anti-monotony constraint—frequency constraint. However, the other two classes of constraints, monotony constraint and the class of tough constraint, have been considered to be difficult to be processed. For either the Apriori-based method or the projection-based pattern growth method, long patterns are needed to be generated from short patterns. However these two classes of constraints can not be easily incorporated into this generation process. It means that the pruning power of these two constraints is difficult to be exploited.

To deal with monotony constraint based sequential pattern mining, some approaches such as ExAnte[5] and PrefixGrowth[4] have been proposed. ExAnte exploits the pruning power of monotony constraint through a pre-processing step by iteratively pruning infrequent items first and pruning the sequences not satisfying monotony constraints then. PrefixGrowth is proposed on the basis of PrefixSpan. It pushes the monotony constraint into sequential pattern mining process by pruning all the sequences that can not satisfy the monotony constraint in the projected database of a prefix.

Tough constraints are the most difficult ones to be processed and have many different concrete forms. The methods to deal with different forms of tough constraints may also be different. Here we discuss a typical class of tough constraint—MaxGap constraint as an example. A MaxGap constraint is defined only in a sequence database SDB where each itemset in every sequence has a timestamp. It requires that the pattern appear frequently in the sequence database such that timestamp difference between every two adjacent itemsets must be shorter than a given gap. For such constraints, there have been proposed two typical algorithms, cSPADE [6] and CCSM [7]. Algorithm cSPADE is designed on the basis of SPADE and pushes the MaxGap constraint into the mining process. It uses  $(k-1)$ -patterns as the prefix and 2-patterns satisfying the MaxGap constraint to produce candidate  $k$ -sequences. In this way, cSPADE might generate a lot of candidates, and need to store all frequent 2-sequences satisfying the MaxGap constraint. Thus it might destroy the prefix-class equivalence self-inclusion of SPADE, which ensures high locality and low memory requirement.

To overcome the limitation of cSPADE for processing the MaxGap constraint, Salvatore Orlando et al. propose CCSM algorithm which generates a candidate  $k$ -sequence by joining two frequent  $(k-1)$ -sequences satisfying MaxGap constraint. Comparing to the cSPADE algorithm, CCSM reduces the number of candidate sequences during the generation process. At the same time CCSM improves the efficiency of composing  $k$ -sequences through Cache technique.

However, each of the above algorithms is proposed for some special constraint based sequential pattern mining and is difficult to be adapted to process other classes of constraints. In this paper we provide a new general framework called CBPSA<sub>l</sub>gm based on the projection-based pattern growth principal. To deal with various constraints, different strategy can be designed and integrated into the framework to construct effective algorithms.

The rest of this paper is organized as follows: Section 2 will present the general framework for constraint based sequential pattern mining. A strategy for processing monotony constraint will be presented in Section 3. In section 4 a strategy for proc-

essing a typical kind of tough constraint—MaxGap constraint is proposed. The experimental results and analysis are presented in section 5. Finally, we conclude our work in section 6.

## 2 The Overview of Framework for Constraint Based Sequential Pattern Mining

In this paper we present a framework CBPSAlgm for constraint based sequential pattern mining. The framework is constructed on the basis of the projection-based pattern growth method. In the framework the specific constraint based pruning strategy can be proposed to be incorporated and the corresponding effective constraint based sequential pattern mining algorithms can be designed. Before presenting our framework CBPSAlgm, we first give some concepts about the sequence, prefix, projection, suffix and projected-database.

A sequence  $s = \{s_1, s_2, \dots, s_j\}$  is an ordered list of itemsets, where  $s_j$  is an itemset and is also called an element of the sequence. Element  $s_j$  is denoted as  $\langle x_1, x_2, \dots, x_m \rangle$ , where  $x_k$  is an item and all the items in an element are supposed to be sorted in alphabetical order. An item can occur at most once in an element of a sequence, but can occur multiple times in different elements of a sequence. A sequence  $\alpha = \{a_1, a_2, \dots, a_n\}$  is called a subsequence of another sequence  $\beta = \{b_1, b_2, \dots, b_m\}$  and  $\beta$  a super sequence of  $\alpha$ , denoted as  $\alpha \subseteq \beta$ , if there exist integers  $1 \leq j_1 < j_2 < \dots < j_n \leq m$  such that  $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \dots, a_n \subseteq b_{j_n}$ .

Let  $\alpha$  be a sequence  $\{a_1, a_2, \dots, a_n\}$ ,  $\beta$  be a sequence  $\{b_1, b_2, \dots, b_m\}$ , where  $m \leq n$ . Sequence  $\beta$  is called a prefix of  $\alpha$  if and only if (1)  $b_i = a_i$  for  $(i \leq m-1)$ ; (2)  $b_m \subseteq a_m$ ; and (3) all the items in  $(a_m - b_m)$  are alphabetically after those in  $b_m$ .

Given sequences  $\alpha$  and  $\beta$  such that  $\beta$  is a sequence of  $\alpha$ . A subsequence  $\alpha'$  of sequence  $\alpha$  is called a projection of  $\alpha$  w.r.t. prefix  $\beta$  if and only if (1)  $\alpha'$  has prefix  $\beta$  and (2) there exists no proper super sequence  $\alpha''$  of  $\alpha'$  such that  $\alpha''$  is a subsequence of  $\alpha$  and also has prefix  $\beta$ . Let  $\alpha' = \{e_1, e_2, \dots, e_n\}$  be the projection of  $\alpha$  w.r.t. prefix  $\beta = \{e_1, e_2, \dots, e_{m-1}, e_m\}$  ( $m \leq n$ ). Sequence  $\gamma = \{e_m'', e_{m+1}, \dots, e_n\}$  is called the suffix of  $\alpha$  w.r.t. prefix  $\beta$ , denoted as  $\gamma = \alpha / \beta$  [4].

We denote the projected database of a sequence database w.r.t. prefix sequence  $pre$  as  $pre$ -projected database. The  $pre$ -projected database is composed by those non empty suffix of each sequence in the original sequence database w.r.t.  $pre$ . If a sequence is not a super-sequence of  $pre$ , the suffix of this sequence w.r.t.  $pre$  is an empty sequence. All empty suffixes will not be stored in the projected database. In order to improve the memory usage efficiency we just store the corresponding projection position of each nonempty suffix of the original sequence and its sequence id in the  $pre$ -projected database.

In projection-based pattern growth method each different prefix is used to project the sequence database to get the corresponding projected-database. Then frequent items in the projected-database are discovered. The prefix is extended with each discovered frequent item to form a new prefix which is a pattern. The projection-based pattern growth method will generate a new projected database based on the new prefix and discover new patterns recursively.

To constraint based sequential pattern mining, we can push constraints into mining process to prune some items that can not belong to any pattern that satisfies constraints. This may help to reduce the size of projected database and lead to further items pruning. For example, given an item  $i_1$ , suppose that  $i_1$  is not a valid extension item for the prefix  $pre$  in sequence  $s_1$ , i.e., the prefix  $pre$  can not be extended with  $i_1$  in sequence  $s_1$ . While in the sequence  $s_2$ ,  $i_1$  is a valid extension item. Under this circumstance, the  $pre$ 's projected database will only contain the suffix of  $s_2$  w.r.t.  $pre$ . This helps to reduce the size of projected database. Furthermore, because that  $i_1$  cannot get support from sequence  $s_1$ , it may lead to that  $i_1$  can not be a frequent item and thus can be pruned. This will in turn help to further reduce the search space.

Here we give the pseudo code of the framework.

---

```

Input:  sequence database:  $SDB$ , minimum support:  $min\_supp$  and the constraint  $C$ 
Output: all sequential patterns that satisfy the constraint  $C$ .
Call method  $CBPSAlgm(< >, SDB)$ 
1. Call  $GetAllValidFrequentItems(prefix, S|prefix)$ 
2. While( $FrequentItemsSet \neq NULL$ ){
    newPrefix =  $ExtendPrefix(prefix, FrequentItem)$ ;
    if (newPrefix satisfies the Constraint)
         $WriteIntoPatternTbl(newPrefix)$ ;
     $CBPSAlgm(newPrefix, S|newPrefix)$ ;
}

```

---

**Fig. 1.** The pseudocode of  $CBPSAlgm$  framework

The function  $CBPSAlgm$  has two parameters: the first is a frequent pattern as a prefix. At the beginning it is an empty sequence, denoted as  $< >$ . The second parameter is the projected database w.r.t  $prefix$ . Initially it is the sequence database in which all infrequent items have been pruned.  $ExtendPrefix$  function will return a new prefix which is joined by  $prefix$  and a frequent item.

The most important part of the framework is the  $GetAllValidFrequentItems$  function. The function corresponds to the constraint processing strategy in mining process. To different constraint, the corresponding constraint based pruning strategy can be designed for this function and incorporated into the framework  $CBPSAlgm$  to get an efficient sequential pattern mining algorithm.

### 3 The Strategy for Processing Monotony Constraint

Let  $C_M$  be the monotony constraint. To such constraint, it has the property that if a sequence  $\alpha$  does not satisfy  $C_M$ , its super sequence might satisfy  $C_M$ . So we can not use apriori trick way to remove sequence  $\alpha$  from search space.

Presently the typical methods for processing the monotony constraint based sequential pattern mining are  $ExAnte$  and  $PrefixGrowth$ . But  $ExAnte$  can not ensure that

all the discovered patterns after preprocessing step satisfy the  $C_M$ . So we still have to check each discovered pattern. On the other hand, by monotone pruning we risk to lose anti-monotone pruning opportunities given by the removed items [8]. Prefix-Growth just prunes infrequent items and invalid sequences in the projected database  $D(pre)$  when mining extendable frequent items for the prefix  $pre$ . It does not push the pruning operation to each item of valid sequences in the projected database.

By the property of monotony constraint  $C_M$ , a theorem can be given as follows:

**Theorem 1.** Given a projected database  $D(pre)$  of prefix  $pre$ . Suppose that  $s_1$  is a sequence in  $D(pre)$ ,  $i$  is an item of  $s_1$ , and  $\text{suffix}(i)$  is the suffix of  $i$  in  $s_1$ . If  $pre + i + \text{suffix}(i)$  does not satisfy  $C_M$ , where the operator ‘+’ means the join of two sequences, then item  $i$  can be pruned safely.

**Proof.** By the property of  $C_M$ , we know that if  $pre + i + \text{suffix}(i)$  does not satisfy  $C_M$ , any subsequence of  $pre + i + \text{suffix}(i)$  can not satisfy  $C_M$  either. Therefore  $i$  can not be a valid extendable frequent item of  $pre$ . If not so, after extending  $pre$  by  $i$  we get a new prefix  $new\_pre = pre + i$ . The projected database of  $new\_pre$  must contain sequence  $s_2 = \text{suffix}(i)$ . Because  $new\_pre + s_2$  can not satisfy  $C_M$ ,  $s_2$  should be pruned and  $new\_pre$  should not be generated either. Thus  $i$  can be pruned safely.  $\square$

Based on the theorem above, we implement a strategy for processing constraint  $C_M$  in function *GetAllValidFrequentItems*, and incorporate it into *CBPSA* for mining monotony constraint based sequential patterns.

---

```

GetAllValidFrequentItems(prefix, S|prefix) {
  for each sequence in the S|prefix s {
    if (prefix satisfies the  $C_M$ ) {
      Count all the items in s;
    }
    else{
      for (int i=seq_len; i>0; --i) {
        if (pre + seq[i] + suffix(seq[i]) satisfies  $C_M$ )
          break;
      }
      for (; i>0; --i)
        Count seq[i];
    }
  }
  return all frequent items;
}

```

---

**Fig. 2.** The pseudocode of *GetAllValidFrequentItems* function

Fig. 2 presents the pseudo code of the *GetAllValidFrequentItems* function. Here we give an example to illustrate the use of the above strategy to prune invalid items based on monotony constraint. Given a sequence  $s = \{a, c, \langle b, d \rangle, e\}$ , suppose that each item of  $s$  has a value,  $a.value = 3$ ,  $b.value = 2$ ,  $c.value = 4$ ,  $d.value = 2$  and  $e.value = 1$ .

Let monotony constrain  $C_M$  be defined as  $\{\text{Sum}(\text{pattern}) \geq 8\}$  and prefix  $pre = \{a\}$ . Function *GetAllValidFrequentItems* will check each item of the suffix of  $s_1$  w.r.t.  $pre$  from the end to the beginning. For item  $e$ ,  $\text{suffix}(e) = \{\}$ , and  $pre + e + \text{suffix}(e) = \{a, e\}$  does not satisfy  $C_M$ . So item  $e$  can be pruned. As the same reason, item  $d$  can also be pruned. Thus, only  $c$  and  $b$  are valid extendable items for  $pre$ .

#### 4 The Strategy for Processing Tough Constraint

If a constraint is neither monotonic, nor anti-monotonic, nor succinct, we called it a tough constraint. When a sequence satisfies a tough constraint, we can not sure that whether its subsequence and super sequence satisfy the constraint or not. So it is a hard work for constraint based sequential pattern mining. For example the MaxGap constraint mentioned above is a tough constraint. Formally a MaxGap constraint is in the form of  $\text{Gap}(\alpha) \leq \Delta t$ , where  $\Delta t$  is an integer. A sequence  $\alpha$  satisfies the MaxGap if and only if  $\{ \beta \in \text{SDB} \mid \exists 1 \leq i_1 < \dots < i_{\text{len}(\alpha)} \leq \text{len}(\beta) \text{ s.t. } \alpha[1] \subseteq \beta[i_1], \dots, \alpha[\text{len}(\alpha)] \subseteq \beta[i_{\text{len}(\alpha)}], \text{ for all } 1 < j \leq \text{len}(\alpha), (\beta[i_j].\text{time} - \beta[i_{j-1}].\text{time}) \leq \Delta t \mid \geq \text{min\_supp} \}$ .

As discussed above, the performance of CCSM is better than that of cSPADE for processing the MaxGap constraint. But CCSM is based on GSP method which does not work very well in the condition of low support threshold, so the property of GSP also affects the performance of CCSM algorithm. CCSM needs to spend some time in query processing in the cache when counting the support of a candidate sequential pattern. Along with the further execution of the algorithm, the cache will consume more memory. On the other hand, CCSM algorithm must start from the mining of the set of 2-patterns. This is a very complex preprocessing step.

Based on the CBPSAlgm framework, we can propose an efficient strategy for MaxGap constraint based sequential pattern mining.

By the property of MaxGap constraint, the following theorem can be given:

**Theorem 2.** Let  $s$  be a sequence of SDB,  $D(pre)$  be a projected database of prefix  $pre$ , and  $s_1$  be a suffix of  $s$  w.r.t  $pre$ . An item  $i$  in  $s_1$  is an extendable item of  $pre$  if and only if  $i.\text{timestamp} - pre.\text{last\_timestamp} \leq \text{MaxGap}$ , where  $pre.\text{last\_timestamp}$  is the timestamp of the last item of  $pre$  in  $s$ .

**Proof.** It is obvious from the definition of MaxGap constraint. □

Based on the theorem 2, we can design the strategy shown in Fig. 3 for mining MaxGap constraint based sequential patterns.

From the above strategy we can see that the generated patterns must satisfy the MaxGap constraint as long as the prefix is extended with valid items. Comparing with CCSM our algorithm does not need a special and complex preprocessing step. Each projected database  $D(pre)$  plays the same role as CCSM’s cache. Both of them can help to reduce the search space, but with the projected database we need not manage a global cache. In addition, CBPSAlgm is performed in the depth first search way and

the mining is just performed in each projected database. So it does not need to store all the patterns for mining new candidate sequences like CCSM, and thus requires less memory than CCSM. Furthermore our algorithm is based on the projection-based pattern growth method, it is more efficient in the condition of low support threshold.

---

```

GetAllValidFrequentItems(prefix, S|prefix){
  for each sequence s in the S|prefix {
    for (int i=0; i<len; ++i){
      if (s[i].timestamp ≤ MaxGap)
        count (s[i]);
    }
  }
  return all frequent items;
}

```

---

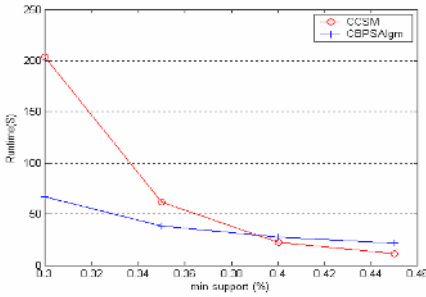
**Fig. 3.** The pseudocode of MaxGap constraint pruning strategy

For example, given a sequence  $s = \{ \langle a, b \rangle, c, d, \langle e, f \rangle, g \}$ , and the prefix  $pre = \{a\}$ , the constraint  $MaxGap = 2$ , the projected sequence in the  $pre$ 's projected database of  $s$  is  $s' = \{ \_b, c, d, \langle e, f \rangle, g \}$ . Based on Theorem 2, we will check all the items in  $s'$ . In sequence  $s$ , the item  $\_b$  and the last item of  $pre$  are in the same transaction, so in  $s'$   $\_b.timestamp - pre.last\_timestamp = 0$ . The timestamp of following transactions in  $s'$  are increased by 1 in turn. At last the valid extendable items are  $\{ \_b, c, d \}$ .

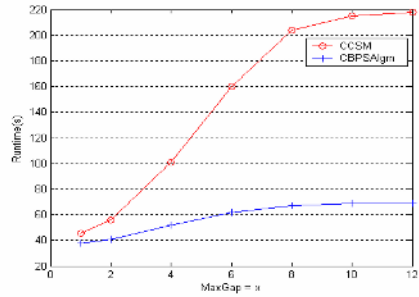
## 5 Experiment and Analysis

In this section we will perform some experiments to evaluate the performance of the algorithms base on the *CBPSAlg*m framework and corresponding pruning strategies. We use the IBM data generator[9] to generate synthetic data for evaluation. The experiment environment is: PIV 2G, 256MB memory and Windows-2000 Professional OS. We compare our algorithms with CCSM, ExAnte, PrefixGrowth algorithms. Our *CBPSAlg*m framework is developed in Microsoft Visual C++ 6.0 and database system is Microsoft SQL Server 2000. In order to ensure the impartiality of the evaluation, we have done some preprocessing work before mining.

Fig. 4 provides the comparison of the efficiency of CCSM and *CBPSAlg*m with a fixed MaxGap constraint and variable minimum support values. The dataset is C10T5S4I3.5D100k and the MaxGap = 8. From the figure we can conclude that in the condition of low minimum support CCSM does not work very well due to that CCSM is based on the GSP method which can not work effectively for pattern mining with low minimum support. The *CBPSAlg*m can work more effectively. With the increase of the minimum support, CCSM and *CBPSAlg*m have almost the same performance.

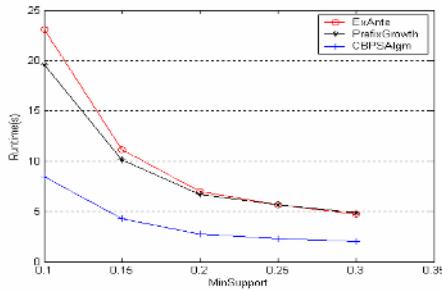


**Fig. 4.** Performance of CCSM and CBPSAlgm with MaxGap = 8



**Fig. 5.** Performance of CCSM and CBPSAlgm with min\_support = 0.3%

Fig. 5 shows the performance of CCSM and *CBPSAlgm* with a fixed minimum support 0.3% and variable MaxGap constraints. We still use the dataset C10T5S4I3.5D100k. From the figure we can see that the number of candidate patterns increase dramatically along with the increase of maxgap, which leads the mining time increasing a lot for CCSM. For *CBPSAlgm*, it partitions all the candidate patterns into different projected databases. In one moment we just need to perform mining task in a projected database of a pattern, which helps to reduce the number of sequences in each iterative step. So it can work more efficiently.



**Fig. 6.** Performance of ExAnte, PrefixGrowth and CBPSAlgm with  $C_M = (\text{Sum}(s) \cdot 300)$

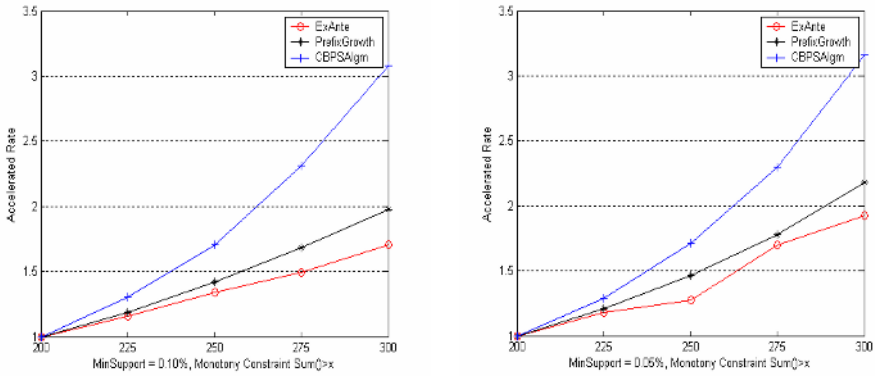
Fig. 6 presents the performance comparison of ExAnte, PrefixGrowth and *CBPSAlgm*. We use a dataset of 50k sequences, the average length of the sequence is 15, and the total number of different items is 5000. Each item has an associated value set as follows:  $\text{item.value} = \text{item} \% 100 + 1$ . The monotony constraint  $C_M$  is  $\text{Sum}(s) \geq 300$  for any pattern  $s$  to be mined.

From Fig. 6 we can see that the efficiency of ExAnte and PrefixGrowth is almost the same while the *CBPSAlgm* works much better. Although the advantage of *CBPSAlgm* reduces along with the increase of minimum support, in the condition of

low minimum support and when  $C_M$  constraint is the primary factor for pruning search space, its efficiency will emerge.

Fig. 7 gives another performance comparison about ExAnte, PrefixGrowth and CBPSAlg with variable monotony constraints when the minimum support is set to be 0.05% and 0.10% respectively. We use the same dataset as Fig. 5-3. The monotony constraints are set as follows:  $\text{Sum}(s) \geq 200$ ,  $\text{Sum}(s) \geq 225$ ,  $\text{Sum}(s) \geq 250$ ,  $\text{Sum}(s) \geq 275$ ,  $\text{Sum}(s) \geq 300$ .

In Fig. 7, the x-axis denotes the different constraint and y-axis denotes the acceleration rate. For example the acceleration rate of an algorithm when the constraint is  $\text{Sum}(s) \geq 250$  can be calculated as follows: acceleration rate = (runtime of the algorithm when  $\text{Sum}(s) \geq 200$ )/(runtime of the algorithm when  $\text{Sum}(s) \geq 250$ ). We can see from the figure that the acceleration rate of CBPSAlg improves much more than those of ExAnte and PrefixGrowth along with the increase of monotony constraints. It is more sensitive to the changes of the constraint than ExAnte and PrefixGrowth.



**Fig. 7.** The performance acceleration rate of ExAnte, PrefixGrowth and CBPSAlg with different monotony constraint when minimum support = 0.05% and 0.10%

From Fig. 6 and Fig. 7 we can conclude that either in the condition of fixed  $C_M$  constraint or in the condition of  $C_M$  increase step by step, CBPSAlg works better than ExAnte and PrefixGrowth algorithms do.

## 6 Conclusions and Future Work

This paper presents CBPSAlg, a new algorithm framework for frequent sequential patterns mining in the presence of user-defined constraints. The CBPSAlg framework is based on the principal of projection-based pattern growth method. In each processing iteration of the framework, a set of valid frequent items are selected to extend current patterns and prune invalid items, which can help to reduce the search space and improve the mining efficiency. For different class of constraints, the corresponding strategies are proposed to prune invalid items. Then each strategy can be



incorporated into CBPSAlgm framework for corresponding constraint based sequential patterns mining.

Finally, some synthetical sequence data is generated with the IBM data generator, and the comparison experiments of our method with some existing algorithm, such as the ExAnte, CCSM algorithm have been done. From the experiment results we can conclude that our framework can perform the task of constraint based sequential pattern mining effectively. In the future, with this framework we plan to focus on new strategies design for developing new constraint based algorithms.

## Acknowledgements

This work is supported by the National Grand Fundamental Research 973 Program of China (No. 2003CB317002), and the Nature Science Foundation of Anhui Province (No.050420305).

## References

1. R. Srikant and R. Agrawal. Mining sequential patterns:Generalizations and performance improvements. In Proc.5th Int. Conf. Extending Database Technology (EDBT'96),pages 3–17, Avignon, France, Mar. 1996.
2. M.J. Zaki. Efficient enumeration of frequent sequences. In *7<sup>th</sup> Intl. Conf. Info. and Knowledge Management*, Nov 1998.
3. Jian Pei, Jiawei Han, Behzad Mortazavi-Asl and Helen Pinto. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In Proc. 2001 Int. Conf. Data Engineering (ICDE'01), pages 215-224, Heidelberg, Germany, April 2001.
4. Jian Pei, Jiawei Han and Wei Wang. Mining Sequential Patterns with Constraints in Large Databases. Proc. of the 2002 ACM CIKM Conference, 2002.
5. Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti and Dino Pedreschi. ExAnte: Anticipated Data Reduction in Constrained Patterns Mining. The 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD03), 2003.
6. Mohammed J. Zaki. Sequence mining in categorical domains: incorporating constraints. Conference on Information and Knowledge Management, 2000.
7. Salvatore Orlando, Raffaele Perego, Claudio Silvestri. A new algorithm for gap constrained sequence mining. Symposium on Applied Computing, 2004.
8. Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, Dino Pedreschi: Adaptive Constraint Pushing in Frequent Pattern Mining. PKDD 2003.
9. Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithm for Mining Association Rules. In Proc. 21st Int. Conf. on Very Large Data Bases (VLDB), pages 432--443, Zurich, Switzerland, 1995.

# Hiding Classification Rules for Data Sharing with Privacy Preservation

Juggapong Natwichai, Xue Li, and Maria Orłowska

School of Information Technology and Electrical Engineering,  
The University of Queensland, Brisbane, Australia  
{jpn, xueli, maria}@itee.uq.edu.au

**Abstract.** In this paper, we propose a method of hiding sensitive classification rules from data mining algorithms for categorical datasets. Our approach is to reconstruct a dataset according to the classification rules that have been checked and agreed by the data owner for releasing to data sharing. Unlike the other heuristic modification approaches, firstly, our method classifies a given dataset. Subsequently, a set of classification rules is shown to the data owner to identify the sensitive rules that should be hidden. After that we build a new decision tree that is constituted only non-sensitive rules. Finally, a new dataset is reconstructed. Our experiments show that the sensitive rules can be hidden completely on the reconstructed datasets. While non-sensitive rules are still able to be discovered without any side effect. Moreover, our method can also preserve high usability of reconstructed datasets.

## 1 Introduction

Currently, many efficient data mining algorithms have been proposed. On one hand, these algorithms can be used by data owners to extract useful patterns from collected data. On the other hand, the algorithms can become a threat in privacy issue. They can be used in combination with other techniques to disclose sensitive private data. For example, a mining result on the medical dataset can help re-identifying of individual person, although the dataset seems to be anonymous.

Not only the threat for individual privacy, but the sensitive private patterns should also be aware. In business, although data sharing is useful for business partners to discover global patterns. However, giving datasets to the others without careful consideration can cause the loss of competitive ability. For example, consider the scenario when a credit card company releases credit card approval dataset for a new home loan company. Each record in the dataset is individual applicant. The collected attributes of each applicant in this dataset can be financial status, number of working years at the current company, gender, salary level, living area and range of age. While the class is the approval result. The purpose of the home loan company is to build a classification model to classify their home loan applicants. The dataset must be provided because two companies have different views on each attribute. However, some sensitive patterns

can be discovered from the given dataset. More specifically, the patterns that exist in the dataset can give competitive ability to the others more than data owners expect. For example, it can be used to identify appropriate groups of customers, or even individual person to send advertising mail. It can be done by changing the class label to be the post code of living area. Although it might not be able to do accurately, only narrow down the scope can be consider as privacy threat. Therefore, the privacy of sensitive patterns also needs to be concerned and preserved.

To preserve the privacy of sensitive patterns, obviously, the dataset is needed to be modified. Consequently, the dataset correctness will be destroyed definitely. However, if the overall characteristics of the dataset can be maintained, the dataset is still usable. In other words, the usability of the dataset is also needed to be preserved. Therefore, modification should be done properly. Recently, many works proposed to hide sensitive association rules [1]. Almost all use heuristic approach to modify the datasets directly by support or confident values decreasing.

Compared with association rule mining, classification rule mining seems to be more complicated problem. Instead of existing of items in association rules mining, classification deals with attributes and attribute values. Moreover, instead of association between attribute values, the ability to classify dataset of each attribute is considered. Therefore, in preserving privacy of classification rules, a heuristic approach to modify dataset directly should be designed with a great care. Otherwise, the usability of modified datasets can be lost by side effect of modification. As we will demonstrate that the heuristic approach may be an inappropriate approach for classification rules hiding.

In this paper, we propose a classification rule hiding method for categorical datasets by reconstruction approach. Instead of arbitrary dataset modification, our method reconstructs a new dataset that contains only non-sensitive rules. Additionally, the usability of new dataset is also preserved. In our method, we extract classification rules of an original dataset firstly by a rule-based classification algorithm. Subsequently, set of non-sensitive classification rules is used to build a decision tree by our algorithm. Finally, a new dataset is reconstructed from the decision tree.

The rest of this paper is organized as follows. Section 2 provides a review of related work. Our proposed approach is shown in Section 3. The experiments and results are brought up in Section 4. Finally, Section 5 gives the conclusion.

## 2 Related Work

Generally, the privacy problem of individual person can be addressed by using some well-known database techniques such as security view management. Statistical security-control is another approach [2], Noise values are added into an original dataset to preserve the privacy, while the correctness of some aggregated values e.g. mean or variance are still preserved.

However, privacy preserving data mining (PPDM) is a different issue. In PPDM, data mining algorithms are also considered. Moreover, not only the pri-

vacy of the individuality is concerned, but also the sensitive patterns. The available approaches can be categorized into a few different groups such as heuristic-based, cryptography-based and reconstruction-based techniques [1]. For association rules privacy preserving, most works tackled the problem by using heuristic approaches [3,4]. Selected values in the dataset are changed to decrease support and/or confident values of sensitive rules. The rules will be hidden successfully if their support and/or confident values are less than specific thresholds. Regarding the classification tasks, most of the research works focus on preserving privacy of individuality [5,6].

Our reconstruction-based approach for classification problem is motivated by a reconstruction-based approach for association rules privacy preserving [7,8]. These works preserve the privacy by firstly extracting selected characteristics of the datasets. The preserving process is done on the characteristics, following by reconstruction of new dataset. The approach of dataset reconstruction has advantageous over the heuristic data modification approaches since it hardly introduces side-effect [1].

### 3 Privacy Preserving in Mining Classification Rules

#### 3.1 Drawbacks of Heuristic Modification Method

In this section, we demonstrate drawbacks of heuristic modification method by examples. A credit card approval sample dataset is shown in Table 1. Every record represents a single person who applied for credit card. The categorical dataset consists of four attributes : the number of years at current work, the marriage status , the gender of applicant, and an attribute of whether the name is on a “black list” or not. Finally, each class label is an approval result. For rule hiding demonstration, firstly, we use a classification algorithm (C4.5 [9]) to obtain a whole set of classification rules. The set of rules is shown in Table 2.

Suppose that the owner of the dataset wants to hide the rule: “**# years at current work = medium & black list = yes**  $\rightarrow$  **approval result = NO**”. The easiest heuristic method (in terms of association rule mining) is to decrease the confidence of the rule. This can be done by alternating values in the right hand side, the class, of the corresponding records. In classification context, it is decreasing of ability to classify datasets. In this case, corresponding records are record number 6 and 14. Suppose that the sixth record is chosen. Subsequently, its class label is changed to **YES**. For checking whether the hiding successes, the dataset has to be classified again. The set of rules on modified dataset is listed in Table 3.

From the result, it seems that the sensitive rule has been hidden successfully. However, there are some differences between the original and the modified set of rules. Some non-sensitive rules are lost e.g. the second, the third and the fifth rules of original dataset. Moreover, some insignificant patterns also become significant. For example, in the original set of rules, there is no rule likes the second, the third and the fourth rules of new set of rules. This result occurs because most of classification algorithms generate the rules according to the

**Table 1.** Original credit card approval dataset

Record No.	# years at current work	marriage status	gender	black list	approval result
1	short	married	female	no	NO
2	short	married	female	yes	NO
3	long	married	female	no	YES
4	medium	divorce	female	no	YES
5	medium	single	male	no	YES
6	medium	single	male	yes	NO
7	long	single	male	yes	YES
8	short	divorce	female	no	NO
9	short	single	male	no	YES
10	medium	divorce	male	no	YES
11	short	divorce	male	yes	YES
12	long	divorce	female	yes	YES
13	long	married	male	no	YES
14	medium	divorce	female	yes	NO

**Table 2.** Original credit card classification rules

Rule No.	Antecedence	Class
1	# years at current work = short $\wedge$ gender = female	NO
2	# years at current work = short $\wedge$ gender = male	YES
3	# years at current work = long	YES
4	# years at current work = medium $\wedge$ black list = yes	YES
5	# years at current work = medium $\wedge$ black list = no	NO

datasets classifying ability of each attribute. An arbitrary modification of some data may effects the ability unintentionally.

The worst case of heuristic modification is when the owner wants to hide the sensitive rules that contains the attribute with the highest ability to classify the datasets e.g. root node of decision tree. For example, the owner wants to hide rule: “# years at current work = long  $\rightarrow$  approval result = YES”. Obviously, there are four corresponding records: the third, the seventh, the twelfth and the thirteenth records. To hide the sensitive rule, assume that the third record is chosen, its class label is changed to **NO**. Table 4 shows the classification result on modified dataset. As we expected, the set of rules is substantially different from the original set of rules.

Obviously, the sensitive rule hiding by dataset modification would impact derived rules significantly. The side effect seems to be uncontrollable. Moreover, the usability of modified datasets is decreased enormously. Therefore, we purpose a radically different way for hiding sensitive classification rules by reconstruction-based approach. Rather than modification of the datasets for changing knowledge, our approach focus on knowledge controlling. Our result datasets may look different from the original. However, theirs characteristics are still preserved, both knowledge and usability.

**Table 3.** Modified credit card classification rules (The sixth record has been modified)

Rule No.	Antecedence	Class
1	# years at current work = short $\wedge$ gender = female	NO
2	# years at current work = long $\wedge$ gender = female	YES
3	# years at current work = medium $\wedge$ gender = female	YES
4	gender = male	YES

**Table 4.** Modified credit card classification rules (The third record has been modified)

Rule No.	Antecedent	Class
1	gender = female	NO
2	gender = male	YES

### 3.2 Problem Statement

Given a dataset  $D$ , a set of classes  $C$ , a set of classification rules  $R$  over  $D$ , and also  $R' \subset R$ ,  $R'$  is a set of sensitive rules, find a dataset  $D'$  such that there exists only a set of rules  $R - R'$  can be derived.

### 3.3 Dataset Reconstruction Method

Our approach starts with classifying original dataset by rule-based classification algorithms e.g. RIPPER [10]. After a set of classification rules is extracted, the owner can identify the sensitive rules. The remaining non-sensitive rules are considered as characteristics of the dataset. Therefore, they are used to build a dataset generator, a decision tree, by our algorithm. Obviously, a number of used rules effects amount of the characteristics to be preserved. So, the unpruned classification rules, less significant rules, can be used in the decision tree building algorithm. Our approach excludes the set of sensitive rules in the algorithm. Therefore, there is no such directly derivable rule in the reconstructed datasets. Finally, a non-sensitive dataset is reconstructed at the same number as the original dataset by the decision tree. In this step, we modify a data generator from the Very Fast Machine Learning toolkit (VFML) [11] for our purpose. Generally, VFML data generator generates a dataset based on concept of an input decision tree. Therefore, we replace the randomized decision tree generator in VFML by the decision tree from the previous step. Each record is built and assigned each attribute value with uniform probability. Then, it is induced through the respected path in the decision tree. Finally, a class label is assigned to the record with the terminal node of the tree.

Using uniform probability data generator provides an advantage to our approach. Obviously, the number of reconstructed records in each path of the trees can be estimated. For example, if a binary attribute "gender" is chosen as the root of a tree, approximated half of reconstructed records will have "male" attribute value, otherwise "female".

**Table 5.** Decision tree building algorithm

Inputs:  $R$  is set of classification rules.  
 $R'$  is set of sensitive rules.  
Outputs:  $DT$  is a decision tree.

---

**While** there is any rule to be induced **do**  
  select a rule  $r$  from  $R - R'$  to be induced order by the classifying ability.  
  **While** the number of approximated reconstructed records does not exceed  
  the number of records that is classified by rule  $r$  in the original dataset  
    **While** the rule  $r$  is not induced completely **do**  
      select the least common attribute in  $r$ ,  
      put selected attribute as non-terminal node  
      of  $DT$ .  
    **End while.**  
    Assign a class for the selected rule.  
  **End while.**  
**End while.**

---

The decision tree building algorithm is shown in Table 5. In the algorithm, each non-sensitive rule is put in a decision tree one by one. The ordering of rules selection is based on their ability to classify original dataset. When any rule is put earlier, it will be in the higher level of the tree. With a uniform probability characteristic of the dataset generator, a rule that appears in the higher level will be used to generate more records. This can help maintaining similarity between original and reconstructed datasets.

For each selected rule, all of its attributes will become a node of the decision tree. The least common attribute among set of all rules is chosen firstly. This can provide many options to induce the trees by allowing a rule to be reflected on many paths of the tree. A number of paths effects the ability to classify the dataset of the each rule. Therefore, we can obtain the most similar dataset in term of usability by controlling the numbers.

Regarding the complexity, this algorithm has  $O(mn)$  time complexity, where  $m$  is the number of non-sensitive rules and  $n$  is the number of attributes of a given dataset.

## 4 Experiments and Results

Two real-life datasets, Credit Card Applicants Approval and 1984 United States Congressional Voting Records datasets from UCI Repository were used in our experiments. For the first dataset, the continuous attributes were transformed to categorical attributes. The number of records is 690 on 15 attributes. While, the voting dataset contains 435 records on 16 attributes.

In the experiments, two rule based classification algorithms: RIPPER and C4.5 Rule were selected. The numbers of classification rules by RIPPER of credit

card and voting dataset were 5 and 4 respectively. C4.5 Rule could also discover the same numbers of rules. 26 and 21 unpruned classification rules were discovered on credit card dataset by RIPPER and C4.5 Rule respectively. While both of them could discover 9 unpruned classification rules on voting dataset. For each experiment, two classification algorithms are used. After a set of classification rules is generated by the first algorithm, some random rules are selected as the sensitive rules. The set of remaining non-sensitive rules are used to build a decision tree by our algorithm. Subsequently, the tree is used to generate a new non-sensitive reconstructed dataset. Finally, the second classification algorithm is used to evaluate the reconstructed dataset. In the experiments, the first and second classification algorithms can be both the same or different algorithms. In our experiments, both single and many rules hiding were investigated.

#### 4.1 Evaluation Metrics

There are three metrics for evaluation. Firstly, the privacy issue must be considered. More specifically, the existing of sensitive rules is considered from the entire set of rules discovered by the second algorithm. Secondly, the side effect from the hiding approach is considered. There are two main metrics to evaluate the side effect: a number of ghost rules and false-drop rules. Ghost rules are the rules that are not sensitive rules and do not exist in the original dataset, but reconstructed dataset. On the contrary, false-drop rules are the non-sensitive rules that do not exist in the reconstructed dataset, but original dataset. These two numbers can also be seen when the second classification algorithm is used. Obviously, these numbers should be kept minimal.

The last metric is the usability of the reconstructed dataset. Because the released dataset are usually used to build the classification model. Therefore, the ability to classify datasets of each attribute should be measured as the usability metric. In the experiments, the gain ratio [9] is used to served our propose. We measure the percentages of gain ratio variations between the original and reconstructed dataset with Equation 1.

$$V = \sqrt{\frac{\sum_{i=1}^n \left(\frac{o_i - r_i}{o_i}\right)^2}{n}} \times 100 \quad (1)$$

where  $o_i$  and  $r_i$  are gain ratios for the  $i$ th attribute on the original and reconstructed datasets. While  $n$  is the number of entire attributes.

In order to evaluate the usability of our decision tree building algorithm, another algorithm has been developed to be compared. It is almost the same as the algorithm in Table 5, but each rule will be put in the tree as many as possible. With this algorithm, the impact of controlling a number of paths for each rule can be investigated. In the experiment, the compared algorithm is called "ALL" algorithm, While our purposed algorithm is called "CONTROLLED".



**Table 6.** Results of single rule hiding on credit card and voting datasets

			Second Algorithm					
			C4.5 Rule			RIPPER		
			Discovered sensitive rules	False drop rule	Ghost rules	Discovered non-sensitive rules	False drop rule	Ghost rules
First Algorithm	Dataset	Used rules						
C4.5 Rule	Credit card	4	0	0	0	3	1	0
		5	0	0	0	4	0	0
		7	0	0	0	4	0	0
		10	0	0	0	4	0	0
		15	0	0	0	4	0	0
		20	0	0	0	4	0	0
	Voting	3	0	0	0	2	1	0
		4	0	0	0	2	1	0
		6	0	0	0	3	0	0
		8	0	0	0	3	0	0
RIPPER	Credit card	4	0	1	0	4	0	0
		5	0	1	0	4	0	0
		7	0	0	0	4	0	0
		10	0	0	0	4	0	0
		15	0	0	0	4	0	0
		20	0	0	0	4	0	0
		25	0	0	0	4	0	0
	Voting	3	0	1	0	3	0	0
		4	0	1	0	3	0	0
		6	0	1	0	3	0	0
		8	0	0	0	3	0	0

## 4.2 Experimental Results and Discussion

The experimental results of single rule hiding are presented in Table 6. From the results, our approach can hide sensitive rules successfully. There is no discovered sensitive rules in any experiments even we used only a set of pruned rules to build the decision trees. Moreover, side effect in term of ghost and false drop rules was hardly found. Even when the first and the second classification algorithms were different, we were able to avoid the side effect successfully by using all unpruned rules to build the decision trees. Remarkably, the side effect could be found in the voting dataset more than the credit card dataset. It means that our approach can hide sensitive patterns in datasets with more discoverable knowledge better than the less one.

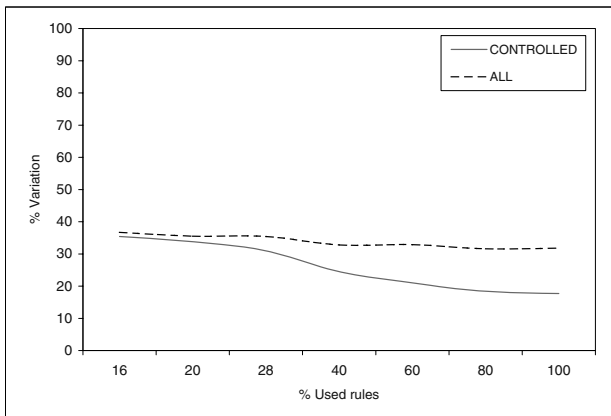
Table 7 shows the experiment results when many sensitive rules were selected to be hidden. In this experiment, RIPPER was used as the first and the second classification algorithms. All unpruned rules were used in the reconstruction process. Obviously, our approach can hide the sensitive rules, and avoid the side effect successfully.

**Table 7.** Results of multi-rules hiding on credit card and voting datasets

Dataset	Hidden rule	Remained non-sensitive rules	Discovered sensitive rules	False drop rules	Ghost rules
Credit card	1	4	0	0	0
	2	3	0	0	0
	3	2	0	0	0
Voting	1	3	0	0	0
	2	2	0	0	0

The Figure 1 shows the usability on reconstructed datasets. In this experiment, RIPPER was used as the first and the second classification algorithms. The percentages of gain ratio variations by numbers of used rules are shown. At 16% of used rules, only pruned classification rules were used, while all unpruned rules were used at 100%.

Generally, the variation decreases when more rules are used in both algorithms. Obviously, our purposed decision tree building algorithm (CONTROLLED) can be used in reconstruction process much more better than the compared algorithm (ALL). Compared with ALL algorithm, controlling a number of paths for each rule in our purposed algorithm can reduce gain ratio variations efficiently. It means that our algorithm can also preserve the usability as well as the privacy.



**Fig. 1.** The credit card datasets usability

## 5 Conclusion

In this paper, we proposed a method of preserving privacy of classification rules of categorical datasets. We can hide sensitive rules by reconstructing a new

dataset which is still similar to the original dataset in terms of knowledge, except the sensitive part. Additionally, our approach can archive high usability of reconstructed datasets. We found that the difference in original and reconstructed datasets can be reduced when we have a large number of rules. In our future work, the efficiency of the approach will be considered.

## Acknowledgments

The work reported in this paper was funded in part by the Australian Research Council - Discovery Project Grant (ARC DP0558879).

## References

1. Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. *SIGMOD Rec.* **33** (2004) 50–57
2. Domingo-Ferrer, J., Torra, V., eds.: *Privacy in Statistical Databases*. Volume 3050 of LNCS. Springer, Berlin Heidelberg (2004)
3. Verykios, V.S., Elmagarmid, A.K., Bertino, E., Saygin, Y., Dasseni, E.: Association rule hiding. *IEEE Transactions on Data and Knowledge Engineering* **16** (2004) 434–447
4. Oliveira, S.R.M., Zaiane, O.R.: Algorithms for balancing privacy and knowledge discovery in association rule mining. In: *7th International Database Engineering and Applications Symposium*, IEEE Computer Society (2003) 54–65
5. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, ACM Press (2000) 439–450
6. Islam, M.Z., Brankovic, L.: A framework for privacy preserving classification in data mining. In: *Proceedings of the 2nd workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation*, Australian Computer Society, Inc. (2004) 163–168
7. Rizvi, S., Haritsa, J.: Maintaining data privacy in association rule mining. In: *Proceedings of the 28th Conference on Very Large Data Base.* (2002) 682–693
8. Chen, X., Orlowska, M., Li, X.: A new framework of privacy preserving data sharing. In: *Proceedings of 4th IEEE International Workshop on Privacy and Security Aspects of Data Mining*, IEEE Press (2004) 47–56
9. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, USA (1993)
10. Cohen, W.W.: Fast effective rule induction. In *Prieditis, A., Russell, S., eds.: Proc. of the 12th International Conference on Machine Learning*, Tahoe City, CA, United States, Morgan Kaufmann (1995) 115–123
11. Hulten, G., Domingos, P.: VFML – a toolkit for mining high-speed time-changing data streams. (2003)

# Clustering-Based Histograms for Multi-dimensional Data

Filippo Furfaro, Giuseppe M. Mazzeo, and Cristina Sirangelo

DEIS, University of Calabria, 87030 Rende, Italy  
{furfaro, mazzeo, sirangelo}@si.deis.unical.it

**Abstract.** A new technique for constructing multi-dimensional histograms is proposed. This technique first invokes a density-based clustering algorithm to locate dense and sparse regions of the input data. Then the data distribution inside each of these regions is summarized by partitioning it into non-overlapping blocks laid onto a grid. The granularity of this grid is chosen depending on the underlying data distribution: the more homogeneous the data, the coarser the grid. Our approach is compared with state-of-the-art histograms on both synthetic and real-life data and is shown to be more effective.

## 1 Introduction

The need to compress data into synopses of summarized information often arises in many scenarios, where the aim is to retrieve aggregate data efficiently, possibly trading off the computational efficiency with the accuracy of query answers. Selectivity estimation for query optimization in RDBMSs [2,6,10], range query answering in OLAP services [11], statistical and scientific data analysis [8], window query answering in spatial databases [1,9], are examples of application contexts where efficiently aggregating data within specified ranges of the domain is such a crucial issue, that high accuracy in query answers becomes a secondary requirement.

For instance, query optimizers in RDBMSs can build an effective query evaluation plan by estimating the selectivity of intermediate query results: this can be accomplished by retrieving aggregate information on the frequencies of attribute values. Obviously, in building an effective execution plan a fast computation of aggregations is mandatory. Moreover a dramatic precision in evaluating aggregates is not needed, as knowing the order of magnitude of the selectivity of intermediate queries suffices to build an effective execution plan. In particular, given a relation  $R(A_1, \dots, A_d)$ , the selectivity of a query of the form  $q = (v'_1 < R.A_1 < v''_1) \wedge \dots \wedge (v'_d < R.A_d < v''_d)$  (representing the intermediate result of more complex queries) is evaluated by accessing the *joint frequency distribution* [10] associated to  $R$ . The latter can be viewed as a  $d$ -dimensional array  $\mathcal{F}$  whose dimensions represent the attribute domains, and whose cell with coordinates  $\langle v_1, \dots, v_d \rangle$  stores the number of tuples of  $R$  where  $A_1 = v_1, \dots, A_d = v_d$ . The selectivity of the query  $q$  defined above is the answer of the range-sum query  $Q = \text{sum}(\langle [v'_1..v''_1], \dots, [v'_d..v''_d] \rangle)$  posed on  $\mathcal{F}$ , which returns the sum of the frequencies contained in the multidimensional range  $\langle [v'_1..v''_1], \dots, [v'_d..v''_d] \rangle$  of  $\mathcal{F}$ . As the size of  $\mathcal{F}$  is generally very large, evaluating the exact selectivity of  $q$  (i.e. the exact answer of  $Q$ ) can be inefficient.

A widely accepted approach to the problem of providing fast estimates of query selectivities consists in compressing  $\mathcal{F}$  into a lossy synopsis  $\mathcal{F}$ , and then evaluating the selectivity of queries by accessing  $\mathcal{F}$  rather than  $\mathcal{F}$ . Histograms [10] are a well-known approach for compressing the joint frequency distribution. A histogram over  $\mathcal{F}$  is built by partitioning  $\mathcal{F}$  into a number of blocks (called *buckets*), and then storing for each bucket  $b$  the number of tuples in  $R$  whose attributes have values belonging to the range of  $b$ . The selectivity of  $q$  is estimated on the histogram by summing the values stored in the buckets whose boundaries are completely contained inside the range-sum query  $Q$  corresponding to  $q$ , and then by estimating the “contributions” of the buckets which partially overlap the range of  $Q$ . These contributions are evaluated by performing linear interpolation, under the assumption that the data distribution inside each bucket is “homogeneous” (that is, the joint distribution of attribute values underlying  $b$  is uniform).

As expected, on the one hand, querying the histogram rather than  $\mathcal{F}$  reduces the cost of evaluating selectivities (as the histogram size is much less than the original data size); on the other hand, the loss of information due to summarization introduces some approximation. Therefore, a crucial issue when dealing with histograms is finding the partition which provides the “best” accuracy in reconstructing query selectivities.

Existing approaches provide reasonable error rates at low-dimensionality scenarios, but worsen dramatically for higher-dimensionality data. On the one hand, this is somewhat inevitable, since, as dimensionality increases, the size of the data domain grows much more than the number of data points. That is, high-dimensionality data are likely to be much sparser than low-dimensionality ones. This implies that the number of buckets which should be used to effectively approximate data tends to explode as dimensionality increases. For instance, consider two data distributions  $D^2$  (of size  $n^2$ ) and  $D^{10}$  (of size  $n^{10}$ ), where the same number of data points are distributed, respectively, on a two-dimensional and ten-dimensional domain. If we use the same number of buckets to partition  $D^2$  and  $D^{10}$ , buckets of  $D^{10}$  are likely to be much larger in volume than those of  $D^2$ . Therefore, the aggregate information associated to buckets of  $D^{10}$  is less localized than buckets of  $D^2$  (as the aggregate value associated to each bucket is spread onto a larger volume), thus providing a poorer description of the actual data distribution.

On the other hand, the low accuracy in query estimates provided by traditional histograms is also due to the ineffectiveness of the adopted heuristics guiding the histogram construction. That is, traditional techniques for constructing histograms often result in partitions where dense and sparse regions are put together in the same bucket<sup>1</sup>, which yields poor accuracy in describing data. For instance consider the bucket shown in Fig. 1(a), where a dense cluster is put together with a sparse region. As the bucket is summarized by the sum of its values, estimating either  $Q_1$  and  $Q_2$  shown in Fig. 1(b) by performing linear interpolation yields a high error rate, since the total sum is assumed to be homogeneously distributed inside  $b$ . In fact this assumption is far from being true: most of the sum of  $b$  is concentrated in the dense cluster on the right-hand side of  $b$ .

Therefore, it’s our belief that improving the ability of distinguishing dense regions can result in more accurate partitions, as this prevents buckets like that of Fig. 1(a)

<sup>1</sup> In Section 3 we will give an evidence of this, by showing partitions of the data domain generated by traditional techniques.

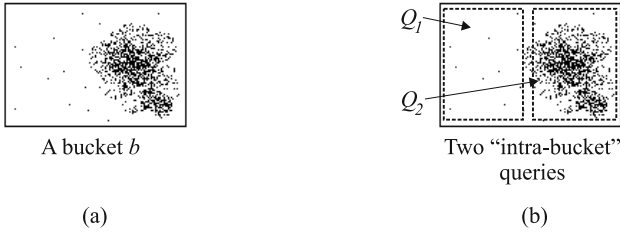


Fig. 1. A non-homogeneous bucket  $b$  and two queries involving  $b$

from being constructed. The problem of searching homogeneous regions is very close to the *data clustering* problem, i.e. the problem of grouping database objects into a set of meaningful classes. This issue has been widely studied in the data mining context, and several algorithms accomplishing data clustering have been proposed. For the sake of brevity we do not provide a classification of existing clustering techniques. The interested reader can find a detailed survey in [7].

This work aims at enhancing the histogram construction by exploiting the capability of clustering techniques to locate dense regions. We define a new technique for constructing multi-dimensional histograms which first invokes a density-based clustering algorithm for partitioning the data into dense and sparse regions, and then further refines this partitioning by adopting a grid-based paradigm.

## 2 CHist: Clustering-Based Histogram

Our technique works in three steps. At the first step clusters of data and outliers (i.e. points which do not belong to any cluster) are located. At the second step, these clusters and the set of outliers are treated as distinct layers, and each layer is summarized by partitioning it according to a grid-based paradigm. At the last step the histogram is constructed by “assembling” all the buckets obtained at the previous step.

The three phases of our approach are described in detail in the following sections. The description of the algorithm is provided by assuming a  $d$ -dimensional data distribution  $D$ .  $D$  will be treated as a multi-dimensional array of integers of size  $n^d$  (without loss of generality the edges of  $D$  are assumed to be of the same size). That is, values of data points of the input distribution are represented into cells of  $D$ . The cells of  $D$  which do not correspond to any data point contain the value 0. A query  $Q$  on  $D$  is specified by a multidimensional range of the domain of  $D$  and its answer is the sum of the values of the cells of  $D$  inside this range.

Any sub-array of  $D$  will be referred to as a *bucket*. The volume of a bucket  $b$  (i.e. the number of cells of the sub-array) will be denoted as  $vol(b)$ , the sum of data point values inside  $b$  as  $sum(b)$ . In order to measure the homogeneity of the data inside a bucket we adopt the SSE (namely *Sum Square Error*), defined as follows:  $SSE(b) = \sum_{\mathbf{i} \in b} (b[\mathbf{i}] - avg(b))^2$ , where:  $avg(b)$  is the average of cell values inside  $b$ ; the expression  $\mathbf{i} \in b$  means that  $\mathbf{i}$  denotes the coordinates of a cell inside  $b$ ;  $b[\mathbf{i}]$  denotes the value of the cell of  $b$  with coordinates  $\mathbf{i}$ . The amount of available storage space for the representation of the histogram will be denoted as  $B$ .

### 2.1 Step I: Clustering Data

In our prototype, we have embedded the clustering algorithm DBSCAN [3] in order to group input data into dense clusters. Indeed, our approach can be viewed as orthogonal to any clustering technique: we have chosen DBSCAN as it is representative of density-based clustering algorithms.

The idea underlying DBSCAN is that points belonging to a dense cluster (except those points lying on the border of the cluster) have a dense neighborhood. A point  $p$  is said to have a dense neighborhood if there are at least  $MinPts$  distinct points whose distance from  $p$  is less than  $Eps$  (both  $Eps$  and  $MinPts$  are parameters crucial for the definition of clusters). Points with a dense neighborhood are said to be *core points*. DBSCAN scans input data searching for core points. Once a core point  $p$  is found, a new cluster  $C$  is created, and both  $p$  and all of its neighbors are grouped into  $C$ . Then  $C$  is recursively expanded by including the neighbors of all core points put in  $C$  at the last step. When  $C$  cannot be further expanded, DBSCAN searches for other core points to start new clusters, until no more core points can be found. At the end of the clustering, points which do not belong to any clusters are classified as *outliers*. Fig. 2 shows an example of clustering obtained by DBSCAN.

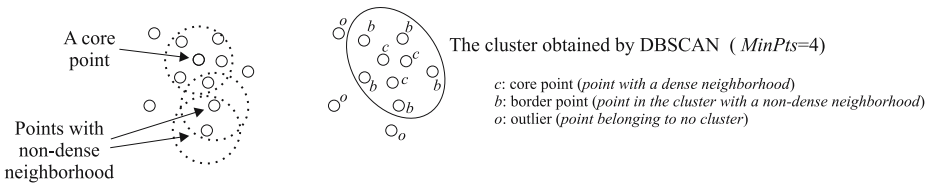


Fig. 2. Running DBSCAN on a set of points

### 2.2 Step II: Summarizing Data into Buckets

At this step the input data distribution is viewed as a superposition of layers. Each layer is either a cluster or the set of outliers. In the following we will denote the layer consisting of outliers as  $L_0$ , and the layers corresponding to dense clusters as  $L_1, \dots, L_c$ .  $L_0$  will be said to be the *outlier layer*, whereas  $L_1, \dots, L_c$  will be said to be *cluster layers*. Each layer is represented by means of its MBR.

The different layers are summarized separately by partitioning their MBRs into buckets. This aims at preventing the construction of buckets where dense and sparse regions are put together, which, as explained before (see Fig. 1), can yield poor accuracy. In more detail, our approach works as follows.

**(Step II.a)** Before summarizing the layers into buckets, possible peaks are located among the set of outliers. In order to detect peaks we use a threshold parameter (namely,  $t$ ) to decide whether an outlier is a peak or not: if the value of an outlier  $o$  is greater than  $t$  times the average value of input points, then  $o$  is a peak, and will be removed from  $L_0$  and stored in detail (this can be viewed as creating buckets containing single points). In our experiments we used the value  $t = 3$ .

**(Step II.b)** The amount of storage space left from the representation of peaks is invested to summarize the clusters and outliers not previously selected. Layers are summarized independently of each other, and the summary of the whole data distribution will be the superimposition of the summaries of all layers.

The summarization of layers is accomplished by a multi-step algorithm which, at each step, summarizes a single layer by partitioning it according to a grid and storing, for each bucket defined by this grid, both its MBR and the sum of its values (obviously, the cells of this grid which do not contain any data point result in an empty MBR which is not stored). The MBRs of buckets obtained from the summarization of cluster layers will be said to be *c-buckets*, whereas the MBRs of the buckets constructed by partitioning  $L_0$  will be said to be *o-buckets*.

Indeed, layer  $L_0$  is processed after the summarization of all the cluster layers. In particular, before summarizing the outlier layer, we scan all outliers to locate those lying onto the range of some c-bucket. Each outlier  $o$  which lies onto some c-bucket is removed from  $L_0$  and “added” to one c-bucket whose range contains the coordinates of  $o$ <sup>2</sup>. This allows us to view c-buckets as “holes” of  $L_0$ , in the sense that, after performing this task, there are no points lying onto the range of some c-bucket which belong to  $L_0$ . As it will be clear in the following, this will be exploited in the physical representation of the histogram to improve its accuracy.

We now describe how the available storage space is used to summarize layers. Let  $B_i$  be the amount of memory which is left from the  $i - 1$  previous summarization steps (at the first step,  $B_1$  is the residual of the initial amount of storage space which is left from the representation of peaks). The portion of  $B_i$  which is invested to summarize  $L_i$  is denoted as  $B(L_i)$  and is computed by comparing the need of being partitioned of  $L_i$  with all remaining layers  $L_{i+1}, \dots, L_c, L_0$ . The need of being partitioned of a layer  $L$  is estimated by computing its SSE (denoted as  $SSE(L)$ ), thus  $B(L_i) = B_i \cdot \frac{SSE(L_i)}{SSE(L_0) + \sum_{j=i+1}^c SSE(L_j)}$ .

We now show how  $B(L_i)$  is exploited to store a partition of  $L_i$  into buckets. The idea is to partition  $L_i$  according to a grid and store, for each cell of the grid containing at least one point, the coordinates of its MBR and the sum of the values occurring in it. The grid on a layer  $L_i$  is constructed as follows.

If we denote as  $W$  the amount of storage space needed to store a bucket<sup>3</sup>, the number of buckets produced by the grid on  $L_i$  can be no more than  $nb = \lfloor \frac{B(L_i)}{W} \rfloor$ . Thus, if  $t_j$  is the number of divisions of the grid along the  $j$ -th dimension of  $L_i$ , it should be  $\prod_{j=1}^d t_j = nb$ .

We partition each edge of the MBR of the layer to be summarized into a number of portions which is proportional to the length of the edge itself. Let  $w_j$  be the length of the edge along the  $j$ -th dimension, and  $t_j$  be the number of divisions performed along the same dimension. Choosing  $t_j = w_j \cdot \sqrt[d]{\frac{nb}{vol(L_i)}}$  (where  $vol(L_i)$  is the volume of

<sup>2</sup> If more than one c-bucket contains  $o$ , one of these c-buckets is randomly selected to incorporate  $o$ . Adding an outlier  $o$  to a c-bucket  $b$  means removing  $o$  from  $L_0$  and adding the value of  $o$  to  $sum(b)$ .

<sup>3</sup> We use  $2 \cdot d$  32-bit words for storing bucket boundaries, and one 32-bit word for storing the sum-aggregate



the MBR of  $L_i$ ), guarantees both that  $\prod t_j = nb$  and that the grid degree along each dimension is chosen by weighting the corresponding edge size. Indeed, this formula can result in non-integer value coefficients  $t_1, \dots, t_d$ .

Therefore we use the following strategy to construct the grid. The degrees of the grid along each dimension are computed progressively, starting from  $t_1$  to  $t_d$ , according to the following scheme:

$$t'_1 = \max\{\lfloor t_1 \rfloor, 1\}; \quad t'_2 = \max\left\{\left\lfloor \frac{t_1 \cdot t_2}{t'_1} \right\rfloor, 1\right\}; \quad \dots \quad t'_d = \max\left\{\left\lfloor \frac{\prod_{j=1}^d t_j}{\prod_{j=1}^{d-1} t'_j} \right\rfloor, 1\right\}.$$

That is, the value of each  $t_j$  is approximated to  $t'_j$  by taking into account the approximations already performed at the  $j - 1$  previous steps.

### 2.3 Step III: Representation of the Histogram

The strategy adopted to compress layers can yield overlapping buckets. In particular, buckets aggregating points of  $L_0$  (the layer consisting of outliers) are likely to be larger than buckets describing clusters. Therefore, several c-buckets  $b_1, \dots, b_k$  can lie onto the range of an o-bucket  $b$ . In this scenario  $b_1, \dots, b_k$  can be viewed as “holes” of  $b$ , as the aggregate information associated to  $b$  does not refer to points contained inside  $b_1, \dots, b_k$ . We now show how this observation can be exploited to make query estimation more accurate. In the following, given an o-bucket  $b$ , the set of c-buckets completely contained into  $b$  will be denoted as  $Holes(b)$ .

Consider the scenario depicted in Fig. 3(a), where the query  $Q_1$  intersects one half of the range associated to the bucket  $b$ . Adopting linear interpolation to estimate  $Q_1$  returns:  $\tilde{Q}_1 = \frac{vol(Q_1 \cap b)}{vol(b)} \cdot sum(b)$ , where  $Q_1 \cap b$  refers to the intersection between the query range and the range of  $b$ . In fact points belonging to the ranges of  $b_1, \dots, b_9$  give no contribution to the value of  $sum(b)$ . Therefore, a more precise estimate for  $Q_1$  is:  $\tilde{Q}_1 = \frac{vol(Q_1 \cap b)}{vol(b) - vol(b_1, \dots, b_9)} \cdot sum(b)$ , where  $vol(b_1, \dots, b_9)$  denotes the volume of the range underlying the buckets  $b_1, \dots, b_9$ . Likewise, the bucket  $b$  should give no contribution to the estimate of the query  $Q_2$  in Fig. 3(b), which lies completely on the range underlying the buckets  $b_1, \dots, b_9$ .

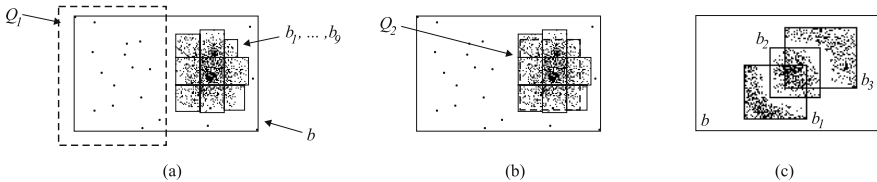
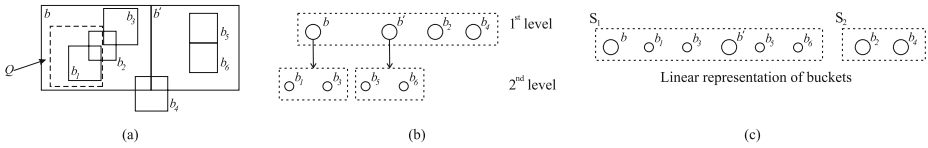


Fig. 3. O-buckets with holes

In the following the number of cells of an o-bucket  $b$  which are not contained in any hole of  $b$  will be said to be the *actual volume* of  $b$ . In the case depicted in Fig. 3(a) evaluating the actual volume of  $b$  can be accomplished efficiently, as  $b_1, \dots, b_9$  do not



**Fig. 4.** Nested representation of buckets

overlap. Indeed also c-buckets inside an o-bucket  $b$  can intersect one another <sup>4</sup>. For instance, in Fig. 3(c) the three buckets  $b_1, b_2, b_3$  inside  $b$  overlap. In this case computing the actual volume of  $b$  requires  $vol(b_1), vol(b_2), vol(b_3), vol(b_1 \cap b_2), vol(b_2 \cap b_3)$  and  $vol(b_1 \cap b_2 \cap b_3)$  to be computed. This computation becomes more and more complex when more buckets intersect in the same region: we need to compute the volumes of all the intersections between 2 holes, 3 holes, and so on. Obviously, this slows down query estimations. Due to this reason, we prefer to estimate the actual volume of an o-bucket  $b$  involved in a query instead of evaluating its exact value: To this end we consider only a maximal subset of  $Holes(b)$  (denoted as  $NOHoles(b)$ ) consisting of non-overlapping c-buckets, thus avoiding intersections between holes to be computed. For instance, in the case depicted in Fig. 4(a) we can estimate the actual volume of  $b$  as  $vol(b) - vol(b_1) - vol(b_3)$ . However we point out that from our experiments on real-life data it turned out that intersections between c-buckets are unlikely to occur.

The adopted representation model partitions buckets into two levels. The buckets at the second level are those belonging to  $NOHoles(b)$  for some  $b$ . The first level consists of all the other buckets.

The physical representation model can be exploited to evaluate query answers efficiently, as it is easy to see that query answers can be estimated by accessing each bucket at most once.

Observe that representing some c-buckets as holes of o-buckets introduces no spatial overhead on the representation of o-buckets. That is, the two-levels organization of buckets can be linearized by representing buckets into two distinct sequences  $S_1, S_2$ . In particular,  $S_1$  contains all o-buckets and their non-overlapping holes: each o-bucket  $b$  is followed by the representation of c-buckets in  $NOHoles(b)$  (see Fig. 4(c)). Thus, locating non-overlapping holes of an o-bucket  $b$  at position  $i$  in this sequence can be accomplished by scanning the positions of the sequence following  $i$ , till either the end of the sequence or an o-bucket having an empty intersection with  $b$  is reached (for instance, the holes  $b_1, b_3$  of  $b$  occur in the sequence between  $b$  and  $b'$ ). Sequence  $S_2$  contains all c-buckets which do not belong to any  $NOHoles(b)$  for any o-bucket  $b$ .

This explains why we do not consider c-buckets which partially overlap o-buckets as holes. For instance, when estimating the query  $Q$  of Fig. 4(a), bucket  $b_4$  is not taken into account to estimate the actual volumes of  $b$  and  $b'$ . Otherwise we should insert into both the representations of  $b$  and  $b'$  a reference to  $b_4$  (which cannot be accomplished by

<sup>4</sup> Although no pair of clusters  $C_1, C_2$  can overlap (otherwise  $C_1, C_2$  would be a unique cluster), MBRs of clusters can overlap (see Fig. 3(c)). Thus partitioning overlapping MBRs can result in overlapping c-buckets.

a sequential physical representation of the histogram), and moreover bucket  $b_4$  should be accessed more than once to estimate queries involving  $b$  and  $b'$ .

### 3 Experimental Results

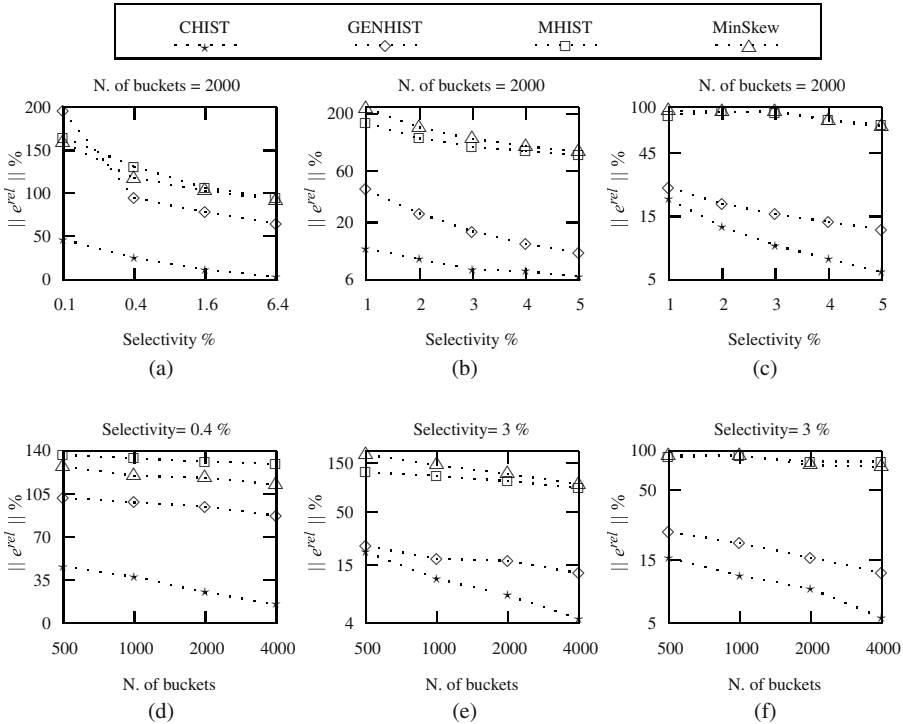
In this section we present some experimental results comparing the accuracy of estimating query selectivities by means of CHIST with state-of-the-art techniques. The accuracy of a histogram  $\tilde{\mathcal{F}}$  built on the joint frequency distribution  $\mathcal{F}$  of an input relation  $R$  is measured by evaluating the average relative error of the estimates obtained by accessing the histogram. Given a query  $q$  on  $R$ , we denote the range-sum query on  $\mathcal{F}$  denoting its selectivity as  $Q$ , and the estimate of  $Q$  evaluated on  $\tilde{\mathcal{F}}$  as  $\tilde{Q}$ . The *relative error* of the estimate  $\tilde{Q}$  is defined as:  $e^{rel} = \frac{|Q - \tilde{Q}|}{\max\{1, Q\}}$ . We performed several experiments both on synthetic and real-life data.

**Synthetic Data.** Our synthetic data are similar to those of [4]. They are generated by creating an empty  $d$ -dimensional array  $D$  of size  $n^d$ , and then by populating  $r$  regions of  $D$  by distributing into each of them a portion of the total sum value  $T$ . The size of the dimensions of each region is randomly chosen between  $l_{min}$  and  $l_{max}$ , and the regions are uniformly distributed in the multi-dimensional array. The total sum  $T$  is partitioned across the  $r$  regions according to a Zipf distribution with parameter  $z$ . To populate each region, we first generate a Zipf distribution whose parameter is randomly chosen between  $z_{min}$  and  $z_{max}$ . Next, we associate these values to the cells in such a way that the closer a cell to the centre of the region, the larger its value. Outside the dense regions, some isolated non-zero values are randomly assigned to the array cells. As explained in [4], data-sets generated by using this strategy well represent many classes of real-life distributions.

**Real Life Data.** We considered a real-life data set which will be referred to as *Forest Cover*. It was obtained from the U.S. Forest Service and is available at the UCI KDD archive site. It consists of 581012 tuples having 54 attributes. Among these, 10 attributes are numerical. As in [5], we considered the tuples projected on these numerical attributes, thus obtaining a 10-dimensional data distribution which will be denoted as  $FC_{10}$ . We projected  $FC_{10}$  on five attributes, thus obtaining a 5-dimensional data distribution which will be denoted as  $FC_5$ .

MHIST [10], MinSkew [1], and GENHIST [5] are the state-of-the-art techniques which were compared with CHIST in our experiments. The comparison was accomplished by considering histograms (with the same number of buckets) obtained by the four techniques. We investigated how the accuracy depends on the number of buckets and on the exact selectivity of the queries. Diagrams (a, d) in Fig. 5 refer to 8-dimensional synthetic data ( $d = 8$ ;  $n = 1000$ ;  $T = 200000$ ;  $r = 200$ ;  $z_{min} = 0.5$ ;  $z_{max} = 2.5$ ;  $l_{min} = 30$ ;  $l_{max} = 200$ ). Diagrams (b, e) in Fig. 5 were obtained on  $FC_5$ , whereas diagrams (c, f) refer to  $FC_{10}$ .

The query workload was constructed by first randomly generating 10000 query centers in the data domain; then, for each of these centers, queries with increasing selectivity were generated by progressively enlarging the query volume, till a selectivity



**Fig. 5.** Accuracy of techniques on 8D synthetic data (a,d),  $FC_5$  (b,e),  $FC_{10}$  (c,f)

threshold is reached (this threshold is 6.4% for synthetic data, and 5% for real-life data). Finally, the results on the accuracy of the answers were grouped by the query selectivity.

From the analysis of the diagrams in Fig. 5 it turns out that, for all the techniques, the accuracy of estimates improves as the number of buckets increases. Likewise, error rates decrease as selectivity increases. This is mainly due to the fact that queries having higher selectivity are likely to have larger volumes: the larger the volume, the more the buckets of the histogram which are completely contained in the query range (such buckets give an exact contribution to the query evaluation).

Diagrams in Fig. 5 show that CHIST outperforms all the other techniques on both synthetic and real-life data.

## 4 Conclusions

We have proposed a new technique for constructing multidimensional histograms providing high accuracy for selectivity estimation. Our technique invokes a density-based clustering algorithm to partition data into dense and sparse regions which are further partitioned according to a grid-based scheme. We compared our technique with the state of the art on both synthetic and real-life data, showing that it yields the best accuracy.

In our prototype we adopted the well-known clustering algorithm DBSCAN, whose execution time turns out to dominate the cost of the histogram construction. In fact DBSCAN is known to provide poor performances (in terms of computational cost) on large data sets with high-dimensionality. Future work will aim at considering different clustering techniques to be embedded into our approach, in order to study how they can be exploited to improve the histogram construction cost while preserving its accuracy.

## References

1. Acharya, S., Poosala, V., Ramaswamy, S., Selectivity estimation in spatial databases, *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD)*, Philadelphia (PA), USA, 1999.
2. Chaudhuri, S., An Overview of Query Optimization in Relational Systems, *Proc. Symposium on Principles of Database Systems (PODS)*, Seattle (WA), USA, 1998.
3. Ester, M., Kriegel, H. P., Sander, J., Xu, X., A density-based algorithm for discovering clusters in large spatial databases with noise, *Proc. 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, Portland (OR), USA, 1996.
4. Garofalakis, M., Gibbons, P.B., Wavelet Synopses with Error Guarantees, *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD)* Madison (WI), USA, 2002.
5. Gunopulos, D., Kollios, G., Tsotras, V. J., Domeniconi, C., Selectivity estimators for multi-dimensional range queries over real attributes, *The VLDB Journal*, Vol. 14(2), April 2005.
6. Ioannidis, Y. E., Poosala, V., Balancing histogram optimality and practicality for query result size estimation, *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD)*, San José (CA), USA, 1995.
7. Kaufman, L., Rousseeuw, P. J., *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, 2005.
8. Korn, F., Johnson, T., Jagadish, H. V., Range Selectivity Estimation for Continuous Attributes, *Proc. 11th International Conference on Scientific and Statistical Database Management (SSDBM)*, Cleveland (OH), USA, 1999.
9. Mamoulis, N., Papadias, D., Selectivity Estimation Of Complex Spatial Queries, *Proc. 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, Redondo Beach (CA), USA, 2001.
10. Poosala, V., Ioannidis, Y. E., Selectivity estimation without the attribute value independence assumption, *Proc. 23rd International Conference on Very Large Data Bases (VLDB)*, Athens, Greece, 1997.
11. Shanmugasundaram, J., Fayyad, U., Bradley, P. S., Compressed data cubes for OLAP aggregate query approximation on continuous dimensions, *Proc. 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Diego (CA), USA, 1999.

# Weighted K-Means for Density-Biased Clustering

Kittisak Kerdprasop<sup>1</sup>, Nittaya Kerdprasop<sup>1</sup>, and Pairote Sattayatham<sup>2</sup>

<sup>1</sup> Data Engineering and Knowledge Discovery Research Unit,  
School of Computer Engineering, Suranaree University of Technology,  
111 University Avenue, Nakhon Ratchasima 30000, Thailand  
{kerdpras, nittaya}@sut.ac.th

<http://www.sut.ac.th/engineering/computer/faculty/nittaya>

<sup>2</sup> School of Mathematics, Suranaree University of Technology  
111 University Avenue, Nakhon Ratchasima 30000, Thailand  
pairote@sut.ac.th

**Abstract.** Clustering is a task of grouping data based on similarity. A popular k-means algorithm groups data by firstly assigning all data points to the closest clusters, then determining the cluster means. The algorithm repeats these two steps until it has converged. We propose a variation called weighted k-means to improve the clustering scalability. To speed up the clustering process, we develop the reservoir-biased sampling as an efficient data reduction technique since it performs a single scan over a data set. Our algorithm has been designed to group data of mixture models. We present an experimental evaluation of the proposed method.

## 1 Introduction

Clustering is the automatic grouping of data based on similarity. There exists a large number of clustering techniques, but the most classical and popular one is the k-means algorithm [1]. Given a data set containing  $n$  objects, k-means partitions these objects into  $k$  groups. Each group is represented by the centroid of the cluster. Once cluster representatives are selected, data objects are assigned to the nearest centers. The algorithm iteratively selects new better representatives and reassigns data objects until no change is made. At this point the algorithm is said to converge. Even though k-means is an effective clustering algorithm, it can sometimes converge to a local optimum. Many methods [2,3,4,5] have been developed to extend the k-means with the common objective of avoiding converging to a bad local optimum. Some methods [6,7,8] search for the best initialization because k-means is known to be sensitive to initial point selection. Other research [9] seeks for the global optimum, at the cost of computation. These researches try to solve the problem of sub-optimal clustering and estimation the appropriate number of clusters [10,11].

Another difficulty of clustering with k-means is that it fails to identify clusters with large variation in sizes since original large clusters tend to be split. Clustering algorithms, such as DBSCAN [12] and CURE [13], have been developed to overcome this kind of difficulty. DBSCAN associates a data point with its density obtained by counting the number of points in a region of radius  $\epsilon$ . The algorithm discovers clusters by connecting regions with sufficient high density, a *MinPts* threshold. DBSCAN

works well in spatial clustering, but it is sensitive to the selection of  $\epsilon$  and *MinPts* and it fails to efficiently discover clusters with highly different densities. CURE algorithm represents a cluster by a set of points, instead of a single representative. Once the representative points are chosen, the algorithm then shrinks these points toward the centroid of the cluster according to a shrinking factor. CURE is an iterative hierarchical-based clustering that works well with discovering cluster of different sizes, but it is sensitive to the selection of representatives and shrinking factor. Moreover, with very large data set, these algorithms degrade considerably.

When clustering massive data set, data reduction is an effective technique to speed up the algorithm. Sampling [14,15,16] is a powerful data reduction paradigm to remedy the inherent complexity of clustering. Uniform random sampling in which every data point has the same probability of being selected has been used extensively in data mining and databases [17,18,19,20]. In the case of data sets with large variation in cluster sizes, density biased sampling [21,22,23] tends to be a better scheme. In density biased sampling, the probability that a data point will be included in the sample is varied by the density of a cluster.

Recent researches [21,22,23] propose several techniques to density biased sampling. Our work also follows this path with a step further on extending the k-means algorithm to work with a weighted sample. We propose an algorithm on density biased sampling based on the reservoir technique and a weighted k-means algorithm to cluster a data sample augmented with weights. The proposed algorithms are explained in Sections 2 and 3, respectively. We present the experimental results in Section 4. The conclusion and our future work are discussed in Section 5.

## 2 Data Reduction Biased by Density

On scalable popular and successful clustering methods such as k-means to work against large data sets, many algorithms like BIRCH [24] and CLARANS [14] employ the sampling technique to minimize data sets. In BIRCH, a CF-tree structure is built after an initial random sampling step. The CF-tree is used as a summarized data structure with statistical representations of space regions stored on leaf nodes. After the phase of CF-tree building, any clustering algorithm can be applied to the leaf nodes. CLARANS also uses uniform sampling to derive initial representative objects for the clusters.

The sampling technique used in these algorithms is uniform random sampling, which assigns every object the same probability of being included in the sample. But many data sets in real life do not follow the uniform distribution scheme. It instead seems to follow the Zipf's distribution [25], for instance, income and population distribution. In these data sets, some areas such as large metropolitan area have much higher population density than the small cities. If all the populations have equal opportunity of being selected as a representative, sparse areas may be missed and not be included in the sample.

### 2.1 Density-Biased Sampling

Density biased sampling [21] is a sampling technique that takes into account the different sizes of the groups. Small groups or sparse regions are assigned higher

probability to be included in the sample than the large groups or dense regions. By biasing the sampling process, small clusters will not be missed or overlooked as outliers.

Recent advancement on clustering very large data sets in which summarized data structure is even too big to fit into main memory, sampling is independently applied to the data set prior to the subsequent clustering phase. Palmer and Faloutsos [21] develop a non-uniform sampling method for clusters that differ very much in size and density. Their method is a generalization of uniform random sampling in that every group of data sets can be assigned different probability of being drawn. When sampling is biased by group density, smaller groups are oversampling, whereas larger groups are under-sampling. Since clusters are not known a priori, Palmer and Faloutsos combine the phase of density information extraction with the biased sampling phase using the hash-based approach. They argue that the inherent collision problem of any hash-based approach will not dramatically degrade the sample.

Nevertheless, their method is significantly affected by noise due to the tendency of oversampling noisy area. Our approach adopts the reservoir technique to eliminate the collision problem of hash-based approach and it is independent on the assumption regarding cluster distribution to avoid the impact of noise.

## 2.2 Density-Biased Reservoir Sampling

We propose a novel approach of adapting reservoir technique [26,27] to perform a density biased sampling on large data sets. Our algorithm can obtain a desired sample through a single data set scan. The proposed method is simpler and requires less resource than the hash-based method [21].

A reservoir-sampling algorithm [26,27] is a simple, unbiased random sampling algorithm for drawing a sample of size  $n$  without replacement from a population of size  $N$  ( $N \geq n$ ). Vitter [26] has developed a one-pass reservoir-sampling algorithm when the population size ( $N$ ) is unknown and cannot be determined efficiently. The term “reservoir” defines a storage area  $j$  ( $j \geq n$ , but mostly  $j = n$ ) to store the potential candidates of the sample. The  $j$  reservoirs are initialized to store the first  $j$  records of the file, that is, all areas of the reservoir pool are initially filled up. Then the algorithm starts scanning the remaining part of the file with a randomly skipping step. The randomly selected record is evaluated as to whether to replace an existing record in the reservoir pool. If it passes the test, the position in the reservoir is also randomly selected. The process stops when the end of file has been reached and the records in the reservoir form a simple random sample of the population. The general procedure of reservoir-sampling algorithm [27,28] is given in Figure 1.

The time complexity of the algorithm is shown [26,27] to be  $O(n(1 + \log(N/n)))$ . In the reservoir-sampling algorithm, each record of the file is assigned a uniform  $(0,1)$  random number. When the reservoir is needed to be updated, each record in the reservoir has the same chance to be replaced by the new record.

Our sampling algorithm generalizes the reservoir scheme for the case of data with different density distribution. In our proposed method, the initial step of partitioning data into groups resembles that of Palmer and Faloutsos [21]. But our subsequent steps are not based on hashing scheme in order to avoid the effect of noise and collision problems.



**Algorithm** Reservoir sampling

Input: a sequential file of  $N$  population

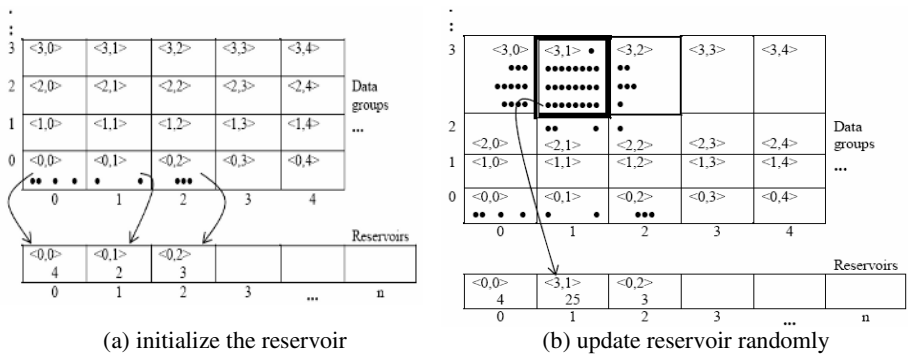
Output: a random sample of size  $n$  ( $n \leq N$ )

- 1) Initialize the reservoir  $X_p, \dots, X_n$  to be the first  $n$  records of the file
- 2) Initialize  $W$  to be the largest value in a sample of size  $n$  from the uniform distribution on the interval  $(0, 1)$
- 3) While not eof do
- 4)     Generate the random variable  $S$  to denote the number of records to be skipped over before a new record can enter the reservoir
- 5)     If (not eof) Then Search for the next potential record to be in the reservoir
- 6)         Else return  $X_p, \dots, X_n$
- 7)     Update  $X$  and  $W$

**Fig. 1.** Reservoir-sampling algorithm

After the initial step of dividing the data space into bins of equal size, the information of the first  $n$  groups are put into the  $n$  reservoirs residing in main memory (see Figure 2a). The collected information includes the number of points in each group and the id of the group.

The algorithm performs a single scan on a data set in a random manner controlled by a random variable  $S$  with the distribution  $W$ . The density biasing (step 7 in Figure 3) is achieved through the consideration of two consecutive data groups. The  $\delta$  threshold is set to detect the cluster edge. Intuitively, a sudden increase or decrease in density with respect to its neighboring area reflects the bordering situation. For example, if the group  $g_i$  contains 30 data points whereas the adjacent group  $g_{i+1}$  contains only 2 data points,  $g_{i+1}$  is highly probably the boundary area of the cluster. With  $\delta$  being set to 20, for instance, the group  $g_i$  is then a candidate to be included in a sample. The  $\epsilon$  value is a threshold to detect noisy and outlier cases. The sparse area is presumably to contain noise or outlier, thus, it should not be put in a sample if its density even combined with the nearby group is below this  $\epsilon$  threshold value.



**Fig. 2.** Density biasing in a reservoir scheme

Figure 2(b) shows the reservoir update for the case of  $\delta$  and  $\epsilon$  values being set to 15 and 5, respectively. The random variable  $S$  is assumed to reach the data group  $\langle 3,1 \rangle$ . On comparison with the adjacent group  $\langle 3,2 \rangle$ , its density is above the threshold values  $\delta$  and  $\epsilon$  (i.e.,  $\| \text{density} \langle 3,1 \rangle - \text{density} \langle 3,2 \rangle \| = 25 - 6 = 19$  and  $\text{density} \langle 3,1 \rangle + \text{density} \langle 3,2 \rangle = 31$ ), thus, the denser group  $\langle 3,1 \rangle$  is a candidate to be included in a sample and is placed in the reservoir pool at a randomly selected position 1. The density-biased sampling proceeds until the skipping variable  $S$  reaches the end of the data groups.

---

**Algorithm** Density-biased reservoir sampling

Input: a data set of  $N$  objects

Output: a density-biased sample of size  $n$  ( $n \leq N$ ) associated with weight  $w$

- 1) Partition data into  $g$  groups (with group-id  $1, 2, \dots, g$ ),  $g \geq n$
- 2) Initialize the reservoir  $X_1, \dots, X_n$  to be the first  $n$   $\langle \text{group-id}, \text{density} \rangle$ -pairs of the data groups
- 3) Set  $W \leftarrow \exp(\log(\text{random}()) / n)$  // initialize  $W$  that will be used in the generation step of a random variable  $S$
- 4) Set  $S \leftarrow \lfloor \log(\text{random}()) / \log(1-W) \rfloor$
- 5) While  $S < g$  do
  - 6) Read data groups  $g_S$  and  $g_{S+1}$  // read two consecutive data groups
  - 7) If  $(\| \text{density}(g_S) - \text{density}(g_{S+1}) \| > \delta)$  OR  $(\text{density}(g_S) + \text{density}(g_{S+1})) > \epsilon$  //  $\delta$  and  $\epsilon$  are predefined density threshold values
 

Then  $X_{\lfloor 1 + \lfloor n * \text{random}() \rfloor \rfloor} \leftarrow \langle \text{group-id}, \text{density} \rangle$  of maximum density  $\{g_S, g_{S+1}\}$  // randomized the reservoir area to be updated
- 8)  $W \leftarrow W * \exp(\log(\text{random}()) / n)$  // update  $W$  for the skipping process
- 9)  $S \leftarrow \lfloor \log(\text{random}()) / \log(1-W) \rfloor$  // generate  $S$  to denote the number of groups to be skipped over

10) Return  $X_1, \dots, X_n$

---

**Fig. 3.** Density-biased reservoir sampling algorithm

### 3 Weighted K-Means Algorithm

The classical k-means algorithm [1] is a fast method to perform clustering. The algorithm consists of a simple re-estimation procedure as outlined as follows. The original  $n$  data points to be clustered are contained in the dataset  $X = \{x_1, \dots, x_n\}$ . The k-means algorithm partitions  $n$  data points into  $K$  sets. The assignment of a data point  $x_i$  to its nearest cluster center  $c_j$  is decided on the basis of the membership function,  $m(c_j|x_i)$ . The function returns either one of the  $\{0,1\}$  values:  $m(c_j|x_i) = 1$  if  $j = \text{argmin}_k \|x_i - c_k\|^2$ ; it is zero, otherwise. The new centroids of clusters can be computed from all data points  $x_i$  in the cluster. The objective function  $J$  of the algorithm is to minimize the sum of error squared,  $J = \sum_{i=1:n} \min_{j \in \{1..k\}} \|x_i - c_j\|^2$ .

**Algorithm** Weighted k-means

Input: a set of  $n$  data points obtained from the density-biased reservoir sampling, and the number of clusters ( $K$ )

Output: centroids of the  $K$  clusters

1) Initialize the  $K$  cluster centers

2) Repeat

Assign each data point to its nearest cluster center according to the membership function,

$$m(c_j|x_i) = \frac{\|x_i - c_j\|^{p-2}}{\sum_{j=1:k} \|x_i - c_j\|^{p-2}}$$

3) For each center  $c_j$ , recompute the cluster center  $c_j$  using the current cluster memberships and weights,

$$c_j = \frac{\sum_{i=1:n} m(c_j|x_i) w(x_i) x_i}{\sum_{i=1:n} m(c_j|x_i) w(x_i)}$$

where  $w(x_i)$  is a weight associated with each data point

4) Until there is no reassignment of data points to new cluster centers

**Fig. 4.** Weighted k-means algorithm

In k-means algorithm, every data point has equal importance in locating the centroid of the cluster. This property does no longer hold in the case of density-biased sample clustering, for which each data point represents varied density in the original data. Therefore, the clustering algorithm has to consider a weight associated with each data point in the computation of cluster centers. The proposed extension to the k-means algorithm is called weighted k-means. Figure 4 outlines the algorithm.

The membership function in the weighted k-means algorithm resembles that of the k-harmonic means algorithm [5]. Zhang [5] also introduces the weight function,  $w(x_i)$ , in his algorithm to accelerate the recomputation of the new centroids in the next iteration. The weight function in our algorithm, however, is introduced for different purpose. It represents the density of the original data points.

## 4 Experiments and Results

We perform two sets of experiments to test the quality of our sampling method, which is the step prior to clustering, and to measure the quality of the weighted k-means algorithm.

### 4.1 Performance of Density-Biased Reservoir Sampling

We evaluate the performance of the proposed reservoir-based density bias sampling method against the hash-based sampling method [21]. The efficiency regarding

memory usage of our reservoir-based sampling method is obviously better than the hash-based method. In the hashing scheme, some amount of memory is needed to store the hashing table in addition to the memory required for storing the drawn sample. Thus, it requires twice the amount of memory comparative to those required by our method.

Effectiveness of the proposed sampling method is examined by measuring the quality of a sample with respect to the number of correctly found clusters. We run clustering using the k-means algorithm. We use a synthetic data generator to generate d-dimensional data sets having  $k$  clusters and  $N$  data points. We vary  $d$  from 2 to 5,  $k$  from 2 to 10, and  $N$  from 5,000 to 100,000.

The measurement *Number of Clusters found* (NC) is the metric defined in [21]. NC is calculated by comparing the distances of the cluster centers found by the clustering algorithm with the true cluster centers. We say that the cluster is found if the calculated distance is less than a predefined threshold (e.g., 0.001).

The results in Figure 5 show the NC when run clustering on various sample sizes with the presence of noise. The reported results are observed from the experiments using 3-dimensional data set having 7 clusters. One cluster contains 50,000 points and the other six clusters contain 500 points. The results obtained from other experiments on data sets with different dimensions, various number of clusters, and varied number of data points are conformed with the one presented in Figure 5, so we omit them for brevity. The experimental results reveal the efficiency of the biased reservoir method especially in the presence of noise.

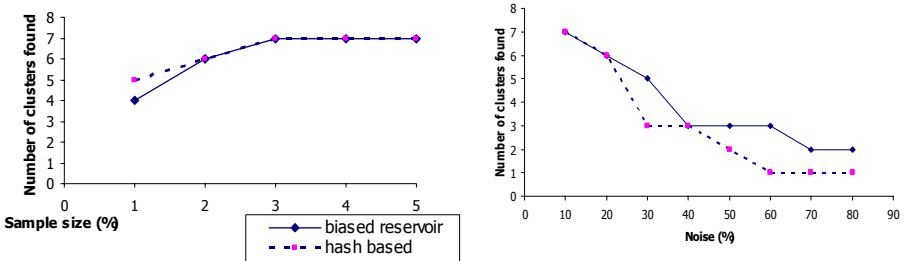


Fig. 5. Finding clusters of 3-dimensional data on various sample sizes, in the presence of noise

### 4.2 Performance of Weighted K-Means Algorithm

We evaluate the quality of the weighted k-means algorithm against the k-means algorithm by using the squared objective function. Lower value of a squared objective function reflects a better quality on clustering. The experiments perform on the syntactic data sets explained in Section 4.1. The initialization step randomly selects data points as initial cluster centroids. We also consider running time of both algorithms.

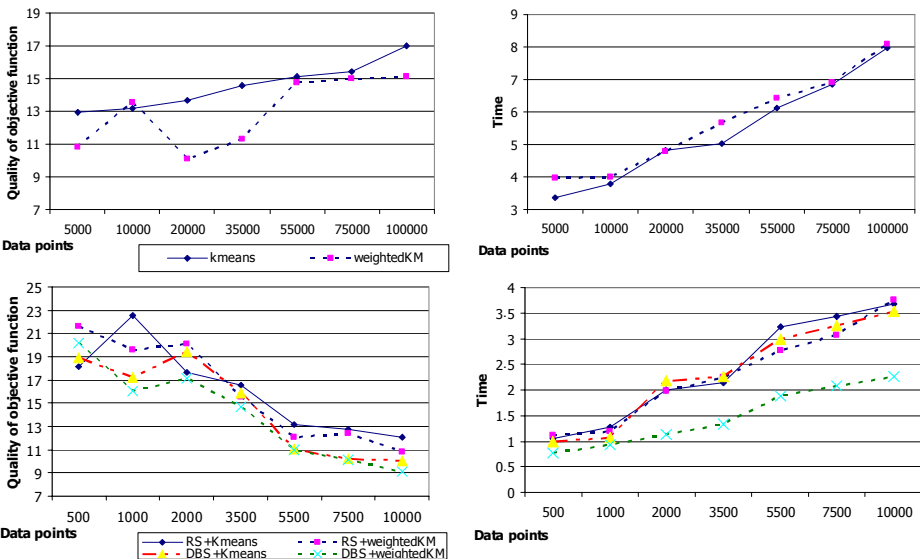
The performance evaluation as shown on top of Figure 6 is obtained from running k-means and weighted k-means algorithms on 3-dimensional data sets of sizes varied from 5000, 10000, 20000, 35000, 55000, 75000, to 100000 data points. The number of clusters is set to be 10. The experiments are performed on the PC with CPU speed

800 MHz, memory 512 MB. Since all data points are used in weighted k-means algorithm, the weight function is set to be 1. The parameter  $p$  in the membership function is set to be 1.3.

The comparison on clustering quality and running time shown at the bottom of Figure 6 reveals the efficiency of running weighted k-means on density-biased sample. The experiments are performed on 10% sample of data with two methods of sampling: simple random sampling (RS) and density-biased reservoir sampling (DBS). The weight function of the weighted k-means algorithm is varied according to the density of the original data.

### 5 Conclusions

The k-means is the simplest and most commonly used clustering algorithm. The simplicity is due to the use of squared error as the stopping criteria, which tends to work well with isolated and compact clusters. Its time complexity depends on the number of data points to be clustered and the number of iteration. We propose a variation of the k-means to better work with a large data set having much difference in cluster density. Our intuition idea is that to cope with massive data set, sampling should be the efficient data reduction method. Since the original data is assumed to be much varied in cluster sizes, density-biased sampling is an appropriate method to preserve the density.



**Fig. 6.** Performance comparison of weighted k-means against k-means (left) and the running time comparison (right), results on top are experiments running on the whole data set while results at the bottom obtained from running on the sample data

We propose a density biased sampling technique based on the reservoir method. The inherent advantage of efficient memory usage in the reservoir scheme is adopted and extended with the additional capability of dealing with data that are much different in density distribution. The proposed technique is designed to lessen the effect of noise as it is the case in the hash-based approach. The experimental results have shown that the proposed method is as good as the hash-based method in discovering correct number of clusters. Our method, moreover, is less sensitive to noisy data even when the percentage of noise is greater than 20.

We also develop the weighted k-means algorithm to better cluster a sample data biased by its density. The results demonstrate the efficiency of the algorithm. The evaluation of the proposed methods on real large databases and the consideration of outliers are our future work.

## Acknowledgements

This research has been supported by grants from the Thailand Research Fund (TRF, MRG4780170), and the National Research Council. The Data Engineering and Knowledge Discovery Research Unit is fully supported by the research grants from Suranaree University of Technology.

## References

1. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In Proc. 5<sup>th</sup> Berkeley Symposium on Mathematical Statistics and Probability, Vol.1. University of California Press (1967) 281-297
2. Hamerly, G., Elkan, C.: Alternatives to the k-means algorithm that find better clusterings. In Proc. 11<sup>th</sup> ACM CIKM Int. Conf. on Information and Knowledge Management (2002) 600-607
3. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
4. Pellog, D., Moore, A.: Accelerating exact k-means algorithms with geometric reasoning. In Proc. 5<sup>th</sup> ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (1999) 277-281
5. Zhang, B.: Generalized k-harmonic means - boosting in unsupervised learning. Technical Report HPL-2000-137. Hewlett-Packard Labs (2000)
6. Bradley, P.S., Fayyad, U.M.: Refining initial points for k-means clustering. In Proc. 15<sup>th</sup> Int. Conf. on Machine Learning (1998) 91-99
7. Meila, M., Heckerman, D.: An experimental comparison of model-based clustering methods. Machine Learning 42(2001) 9-29
8. Pena, J., Lozano, J., Larranaga, P.: An empirical comparison of four initialization methods for the k-means algorithm. Pattern Recognition Letters 20(1999) 1027-1040
9. Likas, A., Vlassis, N., Verbeek, J.: The global k-means clustering algorithm. Technical Report IAS-UVA-01-02. Computer Science Institute, University of Amsterdam, The Netherlands (2001)
10. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In Proc. 17<sup>th</sup> Int. Conf. on Machine Learning (2000) 727-734

11. Sand, P., Moore, A.: Repairing faulty mixture models using density estimation. In Proc. 18<sup>th</sup> Int. Conf. on Machine Learning (2001)
12. Sander, J., Ester, M., Kriegel, H.-P., Xu, X.: Density-based clustering in spatial databases: The algorithm GDBSCAN and its application. *Data Mining and Knowledge Discovery* 2(1998) 169-194
13. Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithm for large databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data (1998) 73-84
14. Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. In Proc. Int. Conf. on Very Large Data Bases (1994) 144-155
15. Zhou, S., Zhou, A., Cao, J., Wen, J., Fan, Y., Hu., Y.: Combining sampling technique with DBSCAN algorithm for clustering large spatial databases. In Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (2000) 169-172
16. Nanopoulos, A., Theodoridis, Y., Manolopoulos, Y.: C<sup>2</sup>P: Clustering based on closest pairs. In Proc. Int. Conf. on Very Large Data Bases (2001) 331-340
17. Singh, G., Rajagopalan, S., Lindsay, B.: Random sampling techniques for space efficient of large data sets. In Proc. ACM SIGMOD Int. Conf. on Management of Data (1999)
18. Toivonen, H.: Sampling large databases for association rules. In Proc. Int. Conf. on Very Large Data Bases (1996) 134-145
19. Thompson, S.K., Seber, G.A.F.: Adaptive Sampling. John Wiley & Sons, New York (1996)
20. Olken, F., Rotem, D.: Sampling from spatial databases. In Proc. Int. Conf. on Data Engineering (1993) 199-208
21. Palmer, C., Faloutsos, C.: Density biased sampling: An improved method for data mining and clustering. In Proc. ACM SIGMOD Int. Conf. on Management of Data (2000) 82-92
22. Nanopoulos, A., Theodoridis, Y., Manolopoulos, Y.: An efficient and effective algorithm for density biased sampling. In Proc. 11<sup>th</sup> Int. Conf. on Information and Knowledge Management (2002) 63-68
23. Kollios, G., Gunopoulos, D., Koudas, N., Berchtold, S.: Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Transactions on Knowledge and Data Engineering* 15(2003) 1-18
24. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data (1996) 103-114
25. Zipf, G.K.: Human Behavior and Principle of Least Effort: An Introduction to Human Ecology. Addison Wesley, Cambridge, MA (1949)
26. Vitter, J.S.: Random sampling with a reservoir. *ACM Transactions on Mathematical Software* 11(1985) 37-57
27. Li, K.-H.: Reservoir-sampling algorithms of time complexity  $O(n(1 + \log(N/n)))$ . *ACM Transactions on Mathematical Software* 20(1994) 481-493
28. Devroye, L.: Non-Uniform Random Variate Generation. Springer-Verlag, New York (1986)

# A New Approach for Cluster Detection for Large Datasets with High Dimensionality

Matthew Gebski and Raymond K. Wong

National ICT Australia and School of Computer Science & Engineering,  
University of New South Wales, Sydney, NSW 2052, Australia

**Abstract.** The study of the use of computers through human computer interfaces (HCI) is essential to improve the productivity in any computer application environment. HCI analysts use a number of techniques to build models that are faithful to actual computer use. A key technique is through eye tracking, in which the region of the screen being examined is recorded in order to determine key areas of use. Clustering techniques allow these regions to be grouped to help facilitate usability analysis. Historically, approaches such as the Expectation Maximization (EM) and K-Means algorithm have performed well. Unfortunately, these approaches require the number of clusters  $k$  to be known beforehand - in many real world situations, this hampers the effectiveness of the analysis of the data. We propose a novel algorithm that is well suited for cluster discovery for HCI data; we do not require the number of clusters to be specified a priori and our approach scales very well for both large datasets and high dimensionality. Experiments have demonstrated that our approach works well for real data from HCI applications.

## 1 Introduction

Usability studies are essential to assess the quality of a user interface. Interface designers may wish to improve the efficacy of an interface or advertisers may wish to determine which part of a webpage receives the most attention. Although there are a number of ways to assess the quality of an interface, a key tool for usability studies is tracking eye movements. In many situations, we may wish to analyze the clusters resulting from the eye movements. These clusters may then be directly matched against the interface components to provide more exact information on various components.

There are a number of domains in which clustering techniques are used, for instance Micro-Array analysis, marketing and finance. Unfortunately, existing approaches require input parameters that often require good cluster analysis techniques in addition to knowledge of the algorithm being used. Moreover, in many cases, domain specific knowledge is also required - this can drastically increase the time between data being collected, and usable knowledge being discovered.

Our aim is to develop an algorithm that can be used immediately at the completion of a eye-tracking or usability testing session. Ideally, we wish for



minimal effort to be required to determine algorithmic parameters that result in high quality clusterings. In this paper, we propose a novel clustering technique that is scalable and particularly suited for the analysis of eye gaze usability data. This enables us to automatically determine the number of clusters, and hence relevant screen components, very quickly.

The remainder of this paper is structured as follows. We begin by examining some of the related work for usability analysis through the use of eye movement and gaze analysis in addition to some of the more prominent clustering techniques. This is followed by the basics of our clustering algorithm which we extend to multiple dimensions. Section 4 provides an experimental evaluation of our approach which we augment with a discussion illustrating the usefulness of our algorithm with respect to HCI tasks.

**Contribution.** Our contribution is a robust clustering algorithm, MCA, that is scalable in terms of both data set size and dimensionality. Unlike other approaches, MCA does not require the number of clusters to be known *a priori*, instead determining the number of clusters based on the data set itself requiring only a single parameter representing a tolerable distance that points may lie from the mean of a cluster. Moreover, we demonstrate empirically that this threshold value falls within a small range for many real world data sets.

## 2 Related Work

Analysis of eye movements has enjoyed favorable attention from the CHI community. Uses of eye movement data includes analysis of web pages [7], [12], determining where attention is drawn during problem solving [11], and providing a basis for user interaction and control [13], [4]. Unfortunately, for many of these problems, the cost of analyzing the gaze data can be cumbersome. Eye movements must be recorded, the data preprocessed by a usability analyst knowledge, discovered in the data such as important screen regions must be identified and mapped to interesting screen components.

There have been a number of clustering algorithms proposed in the data mining literature. Of these, possibly the most notable is the Expectation Maximization (EM) presented as the  $k$ -means algorithm or the similar  $k$ -medoid algorithm. These algorithms begin with a number of seed points representing clusters and assign each point to its nearest seed.  $k$ -means is intuitive, simple to implement and effective for many real world clustering tasks, however  $k$ -means suffers from a number of problems which make it particularly unsuitable for analysis of eye tracking data:

- $k$ , the number of clusters, is a parameter that must be specified beforehand. This ensures that a minimum level of analysis of the data is required before clustering can be performed.
- The choice of the  $k$  seed points greatly influences the final clustering. Being so susceptible to outliers means that the final clustering may be very inaccurate; a single 'true' cluster may be represented as two clusters in the

clustering, while two 'true' clusters may be represented as a single cluster. As medians tend to be influenced to a lesser degree than means by the presence of outliers,  $k$ -medoid is more robust than  $k$ -means, but the actual discrimination of points can still be heavily affected by the initial seed points.

In contrast, hierarchical approaches such as CHAMELEON, [8], assume that all the data is comprised of a series of smaller clusters, which in turn are comprised of smaller clusters (or alternately, all data is a series of smaller clusters that represent a series of larger clusters). A measure of similarity between pairs of clusters is used to determine which should be merged and which should be kept separate. Despite generally resulting in good clusterings, the scalability of hierarchical methods is typically  $O(n^2)$ .

Other approaches include density bases techniques such as DBSCAN/OPTICS which examines the density surrounding each point [1] (the same technique was used for the LOF outlier detection algorithm [2]). Points in high density areas are placed into clusters with other nearby points that are also in high density areas. DBSCAN and OPTICS produce good clusters and requires only one parameter,  $MinPts$ , the number of neighbors of each point to examine when determining the density. Once density information has been collected, the running time of both approaches is linear, however, DBSCAN and OPTICS require the nearest neighbors to be found for each point as part of the clustering process, making high dimensional cluster detection difficult, requiring a running time of  $O(n^2)$ . Additionally, there are a number of categorical and text clustering approaches [6], [10], [9]. Although many of the approaches can be used for real valued data, there is typically a high cost in terms of complexity. Of note is the TURN\* algorithm [5], a parameterless approach which is reasonably efficient. TURN\* calculates the closeness of points at various resolutions and uses this as a measure to decide cluster membership.

### 3 Our Approach

In this section, we propose Mean Clustering Algorithm (MCA). There are two main processes employed by MCA. Firstly, clustering is performed on each dimension. Secondly the results from the clustering of the initial dimension are fed into the clustering process for the subsequence dimension, these steps are then repeated. For clarity, we will begin by considering cluster detection for univariate data, we then extend this approach for an arbitrary number of dimensions.

#### 3.1 Single Dimension Clustering

**Preliminaries.** For the remainder of this paper, we will use  $S_d$  to denote a set of points sorted by dimension  $d$ ,  $S_d[i]$  representing the  $i^{th}$  point in  $d$ .  $S$  is drawn from at least one, although usually many and with varying parameters, Normal distributions. Unless otherwise specified, we will assume that we iterate through  $S$  in ascending order, so that  $S_d[i] < S_d[i+1]$ . A cluster,  $C$ , is an ordered sequence of adjacent, in a given dimension, points.  $C + p$  denotes the assignment of point  $p$  to cluster  $C$  and  $C \oplus p$  represents the temporary assignment of  $p$  to  $C$ .

**Algorithm 1** Clusters a set of points based on one dimension

---

```

CLUSTERDIM ( $S_d$ )
1:  $Clusters = \emptyset$ 
2:  $C' = \emptyset$ 
3:  $Cnt = 0$ 
4: for  $p \in S_d$  do
5:   if  $NumDevs(C' \oplus p, p) < T$  then
6:      $C' + x$ 
7:   else
8:     if  $Cnt > 0$  then
9:       if  $CheckMerge(Clusters[Cnt], C')$  then
10:         $Merge(Clusters[Cnt], C')$ 
11:      if Did not merge then
12:         $Cnt = Cnt + 1$ 
13:         $Clusters = Clusters.add(C')$ 
14:         $AdjustBorrowed(Clusters[Cnt - 1], C')$ 
15:         $C' = \emptyset$ 
16: return return  $Clusters$ 

```

---

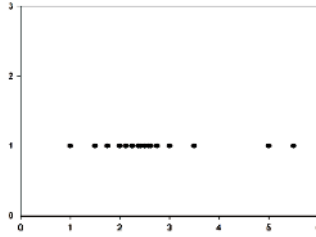
**Approach.** Let us consider a set of points drawn from a single standard normal distribution in one dimension -  $S_1$ . If we were to iterate over these points, keeping track of points observed,  $C'$ , in addition to the mean of the points observed,  $\overline{C'}$ , we would notice that as we 'approached' a cluster  $\overline{C'}$  would begin at the lowest value of  $C'$  and begin to approach 0 we reached the tail the cluster (at approximately 3). Moreover, the standard deviation of the observed points,  $\sigma(C')$ , would begin to approach 1, the true value.

If we were to extend this example, and add another point,  $x$ , say at 10.0 - ten standard deviations from the mean of the original observed points, we would expect that, with a reasonable level of confidence, that  $x$  did not belong to  $C'$ . In fact, if we were to specify a threshold of  $T$  standard deviations, we could classify any points more than  $T * \sigma(C')$  from  $\overline{C'}$  as belonging to different clusters.

For clarity, we have two notions of a 'cluster'. Firstly,  $C$ , a cluster which is complete and will have no more points added to it. The second type of cluster,  $C'$ , represents a group of points that have been considered, but may have more points added at a later stage. We will refer to clusters of the second type as *biased clusters*. We now examine the algorithm for a single dimension as presented in Algorithm 1.

We begin by creating an initial cluster  $C_0'$ , containing the first point of  $S_d$ . The second point is observed and we test to determine whether it should be included in  $C_0'$ . If it should be, then we add it to  $C_0$ . We then repeat this for the third point, fourth point and so on until we begin observing points that should not be included in  $C_0'$ .

At this stage, we consider the remaining, unseen points to have a lower probability of belonging to  $C_0'$  than we are inclined to accept. We then define a new cluster  $C_1'$ . If  $S_d[n]$  is the last point of  $C_0$ , we assign  $S_d[n + 1]$  to be the first



**Fig. 1.** Example set of points in one dimension

point of  $C_1'$ . Following this, we resume testing for membership from  $S_d[n + 2]$  onwards. The process of assigning points to the current cluster until they are outside the acceptable threshold and then creating a new cluster is repeated until all points in  $S_d$  have been assigned.

We illustrate this process with an example. Consider Figure 1, observing the points from left to right and assuming a threshold of  $T = 3$ . Initially, a new, empty cluster,  $C'_0$ , is formed and to which the point at 0 and the point at 1 is assigned. The point at 1.5 is then tested and, as it is within 3 standard deviations of  $\bar{C}'_0$ , it too is added to  $C'_0$ . Points are continually added until we reach the observation at 5.5. As 5.5 is more than the tolerable number of standard deviations from the mean of our cluster, we begin a new cluster and resume testing points. We should note that as we are observing the points from left to right, the mean of a biased cluster will be lower than the mean of the full cluster.

### 3.2 Extension to Multiple Dimensions

Although the one dimensional component forms the crux of MCA, it is not overly useful by itself. Let us imagine a bivariate set of data drawn from three normal distributions; all with a variance of 4, the first with a mean of (2,2), the second with a mean of (10,10) and the third with a mean of (10, 2). In order to determine the structure of these points, we first perform a clustering on the first dimension giving us two clusters. One cluster comprised of the points in the normal at (2,2) while the second cluster will contain both the points in the normal located at (10,10) and (10,2). The resulting clusters are then clustered independently based on the second dimension. The result of clustering the first cluster will simply be (2,2). However, the result of clustering the second cluster in the context of the second dimension will be two clusters, firstly (10,10) and secondly (10,2). Thus resulting in all three clusters being located.

To handle an arbitrary number of dimensions, we begin by taking the points, and projecting them and then clustering them in the first dimension. As each cluster found as a result of this process is separable from the other clusters in the first dimension, we know that they should be kept as distinct groups. Each of the clusters found as a result of the clustering in the first dimension

**Algorithm 2** Clusters points in multiple dimensions

---

```

CLUSTERPOINTS ( $S$ )
1:  $ChildClusters = \emptyset$ 
2: for  $d = 0, 1 \dots num\_dims$  do
3:    $S_d = Load(d)$ 
4:   if  $ChildClusters.empty()$  then
5:      $ChildClusters = ClusterDim(S_d)$ 
6:   else
7:      $Clusters = \emptyset$ 
8:     for  $C \in ChildClusters$  do
9:        $DimPoints = GetDimPoints(C)$ 
10:       $DimClusters = DimClusters + ClusterDim(DimPoints)$ 
11: return  $ChildClusters$ 

```

---

are then processed in the same way. For example, if we to find clusters  $C_1$  and  $C_2$  in the first dimension, we would then process, separately,  $C_1$  in the second dimension and  $C_2$  in the second dimension. Any clusters that are found in the  $n^{th}$  dimension as a result of a cluster from the  $(n - 1)^{th}$  dimension are referred to as *child clusters*. This approach is highlighted in Algorithm 2.

**Identifying Cluster Boundaries.** In the univariate case, when considering points from left to right (ascending order), the 'tail clusters' that occur when ending a cluster will be on the right hand side of the biased clusters. For  $T = 3$ , these tail clusters will contain approximately one percent of the points that should be assigned to the cluster. If we were to instead observe the points from right to left (descending order), the tails would be on the left hand side of the cluster.

Moreover, due to the small number of points that typically make up a tail cluster, it is possible for a tail cluster of a small, dense cluster to be significantly influenced by a large cluster with high variance. This is addressed by performing a *reverse merge* in which we attempt to connect a tail cluster to the cluster to which it should belong (the term *reverse* is used because if we are observing the points in ascending order, the tail cluster will be joined to the preceding points). When we complete the tail cluster, we examine the end of the full cluster - if the points are within the threshold, we merge the tail and full clusters.

**Borrowed Points.** In the basic algorithm, it is possible that some points belonging to a cluster may be further from the mean than permitted by the choice of the threshold value. These points will be assigned, in many cases wrongfully, to the beginning of the adjacent cluster to the right. We refer to this process as *borrowing* (the points themselves are *borrowed*), which can lead to a high level of mis-classification.

We address this issue by calculating the likelihood of the potentially borrowed point belonging to the current cluster, and compare this with the probability of the point belonging to the previous cluster. The test for a point being borrowed is defined as:

---

**Algorithm 3** Adjusts points that are borrowed from the previous cluster

---

ADJUSTBORROWED ( $C_i, C_{i+1}$ )

- 1:  $right\_start = NumDevs(C_i \oplus C_{i+1}[0], C2[0]) < NumDevs(C_{i+1}, C_{i+1}[0])$
  - 2: **while** not  $right\_start$  **do**
  - 3:      $C_i \leftarrow C_i + C_{i+1}[0]$
  - 4:      $C_{i+1} \leftarrow C_{i+1} - C_{i+1}[0]$
  - 5:      $right\_start = NumDevs(C_i \oplus C_{i+1}[0], C2[0]) < NumDevs(C_{i+1}, C_{i+1}[0])$
- 

---

**Algorithm 4** Check to see if two adjacent clusters should have a reverse merge performed

---

CHECKMERGE ( $C_i, C_{i+1}$ )

- 1:  $last = C_i[C_i.size()]$
  - 2: **if**  $NumDevs(C_{i+1} \oplus last, last) < T$  **then**
  - 3:     **return** *true*
  - 4: **else**
  - 5:     **return** *false*
- 

$$IsBorrowed = NumDevs(C_n + C'_{n+1}[0], C'_{n+1}[0]) < NumDevs(C'_{n+1}, C_{n+1}[0])$$

If this point is borrowed, the point is reassigned to the previous, correct, cluster. As we have a new beginning point for the current cluster that may also be a borrowed point, we repeat this test until all borrowed points have been correctly assigned. The reassignment is performed once we begin observing points outside the threshold for the current cluster. Before this stage, the estimate of the mean for the current cluster is heavily biased to such an extent that the probability of points being classified as borrowed is artificially low.

**Accounting for correlation and Obscured Clusters.** In practice, we may find clusters of multivariate data to be heavily correlated. In order to determine clusters for such points, we can perform some form of dimensionality reduction such as Principal Component Analysis (PCA) which creates orthogonal dimensions. Alternately, we can use a different cutting plane based on the covariance of the data. Additionally, in an earlier projection, some clusters may be obscured, but found at a later stage. In order to overcome this, we can rerun the clustering process until no new clusters are found.

## 4 Performance Evaluation

### 4.1 Performance Evaluation of MCA for HCI Data

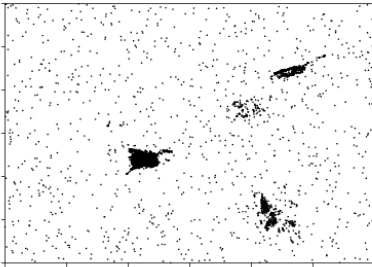
We ran our clustering algorithm on a number of usability data sets, such as the one depicted in Figure 2. In the first part of the figure, we have added noise to make the task harsher; an eyeball test indicates four main clusters in the

**Algorithm 5** Performs a reverse merge

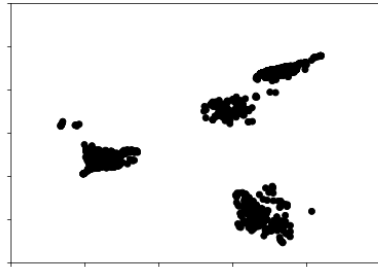
---

 REVERSEMERGE ( $C_i, C_{i+1}$ )

- 1:  $C_{new} = \emptyset$
  - 2:  $C_{new}.start = C_i.start$
  - 3:  $C_{new}.end = C_{i+1}.emd$
  - 4:  $C_{new} = \frac{|C_i|*C_i + |C_{i+1}|*C_{i+1}}{|C_i| + |C_{i+1}|}$
  - 5:  $C_{new}.var = CombineVar(C_i, C_{i+1})$
- 



(a) Eye tracking data with noise



(b) Clean eye tracking data

**Fig. 2.** Sample eye tracking data sets

presence of the added noise. Figure 2b is a clean version of the same data. MCA detects four main clusters for the noisy version of the test, while for the second one, the four that are obvious from the noisy task in addition to two small clusters on the left middle area and an additional small cluster in the lower right. Overall, for real world usability data, our approach performs well except in the presence of a very, very large number of small clusters. In particular, the logical grouping of GUI components allows MCA to perform particularly well. For data sets of this style, MCA will classify the points into one large cluster (although, in many cases, the clusterings produced by k-means is not necessarily usable). The running time required for approximately 40,000 points is under one second.

## 4.2 Performance Evaluation for Synthetic Data

Although eye tracking data is typically one dimensional, we also examine the performance for high dimensional synthetic data. Additionally, we compare MCA against the TURN\* algorithm. Figure 3 demonstrates the scalability of MCA on data over 10 dimensions (we should note that each dimension is sorted). We can see that MCA is linear in performance.

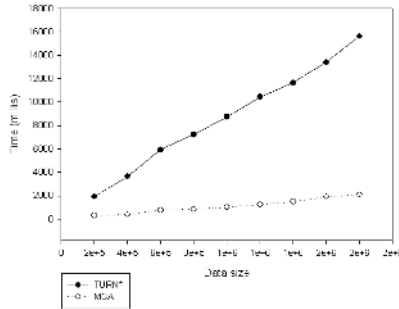


Fig. 3. Performance for MCA vs TURN\*

### 4.3 Scalability Analysis

There are three main steps in the algorithm:

1. Iteration over the points, testing for membership, and assignment of the points to their appropriate clusters. This can be done efficiently by keeping track of the current variance and mean, and updating these when new points are seen. There are numerous ways to do this (some more appropriate than others) as described in Chan and Lewis [3]. As each point only needs to be observed once (in this step), the initial assignment of points to clusters can be performed in  $O(n)$  time and space.
2. Adjusting, borrowed points. The test for borrowed points is only conducted when the cluster is being completed, and the maximum number of points that may be re-assigned would be the number of points in the clusters. As the test for borrowed points is performed in cases where no reverse merge is performed, in the worse case, we will test at most  $O(n)$  points for borrowing. We should note that this  $O(n)$  is per dimension, not per point or cluster and that in practice, the number of tests is typically  $O(k)$ , where  $k$  is the number of clusters.
3. Checking for, and performing, reverse merges. The check for a reverse merge takes  $O(1)$  time as it is simply a membership test. The mean and the new variance of the merged cluster can be calculated in  $O(1)$  time as we are already storing the data required.

For the univariate case, the running time and space is  $O(n)$  if the data is sorted. If the data requires sorting beforehand, the running time would naturally be  $O(n \log n)$ .

For the multivariate case, the number of calls to *ClusterDim* will be equal to the number of clusters,  $Clusters_d$ , at each dimension. However, each call of *ClusterDim* will only process  $\frac{n}{Clusters_d}$  points on average. This means that each dimension will require  $O(n)$  time, irrespective of the number of clusters. As each dimension requires  $O(n)$  time and there are  $d$  dimensions, the total time required will be  $O(nd)$ .



## 5 Discussion

Our clustering approach is very suitable for handling most eye tracking tasks, outputting high quality clustering with only a minimal amount of knowledge required to determine parameters. This makes the usability analyst's job significantly easier, allowing them to focus on analyzing the eye movement data as opposed to the eye movement data. We suggest a framework for usability experiments as follows:

1. Define regions of interest within a screen. Many toolkits and interface design systems make this task easy as widgets tend not to overlap.
2. Perform usability analysis with a user executing pre-defined tasks
3. Take the results of the previous step and generate clusters using MCA
4. Map clusters to the regions defined in the first step.

The advantages of MCA over other approaches are primarily the parameters required and the high scalability allowing the results to be obtained promptly.

## 6 Conclusions

In this paper, we have proposed a new clustering algorithm, MCA, that requires only a single, easy to determine, parameter. MCA, is particularly suited to the task of cluster detection for eye movement data that is often found in HCI analysis. Moreover, the MCA algorithm is scalable and extensible to an arbitrary number of dimensions. The results of our experimental evaluation are very encouraging. MCA provided high quality clusterings in the presence of both noise and varying cluster size. Additionally, in terms of performance, it was very satisfactory, demonstrating the efficacy of MCA for a number of problem areas including the analysis of large databases.

## References

1. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD Conference*, pages 49–60. ACM Press, 1999.
2. M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD Conference*, pages 93–104. ACM, 2000.
3. T. F. Chan and J. G. Lewis. Computing standard deviations: accuracy. *Communications of the ACM*, 22(9):526–531, 1979.
4. D. Fono and R. Vertegaal. Eyewindows: evaluation of eye-controlled zooming windows for focus selection. In *CHI '05: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 151–160, New York, NY, USA, 2005. ACM Press.
5. A. Foss and O. R. Zaïane. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *ICDM*, pages 179–186. IEEE Computer Society, 2002.

6. G. Gan and J. Wu. Subspace clustering for high dimensional categorical data. *SIGKDD Explor. Newsl.*, 6(2):87–94, 2004.
7. J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott, and A. M. Wichansky. Eye tracking in web search tasks: design implications. In *ETRA '02: Proceedings of the symposium on Eye tracking research & applications*, pages 51–58, New York, NY, USA, 2002. ACM Press.
8. G. Karypis, E.-H. S. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
9. K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 658–665, New York, NY, USA, 2004. ACM Press.
10. M. J. Maa-Lpez, M. D. Buenaga, and J. M. Gomez-Hidalgo. Multidocument summarization: An added value to clustering in interactive retrieval. *ACM Trans. Inf. Syst.*, 22(2):215–241, 2004.
11. J. Ou, L. M. Oh, J. Yang, and S. R. Fussell. Effects of task properties, partner actions, and message content on eye gaze patterns in a collaborative task. In *CHI '05: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 231–240, New York, NY, USA, 2005. ACM Press.
12. B. Pan, H. A. Hembrooke, G. K. Gay, L. A. Granka, M. K. Feusner, and J. K. Newman. The determinants of web page viewing behavior: an eye-tracking study. In *ETRA '2004: Proceedings of the Eye tracking research & applications symposium on Eye tracking research & applications*, pages 147–154, New York, NY, USA, 2004. ACM Press.
13. P. Qvarfordt and S. Zhai. Conversing with the user based on eye-gaze patterns. In *CHI '05: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 221–230, New York, NY, USA, 2005. ACM Press.

# Gene Expression Biclustering Using Random Walk Strategies

Fabrizio Angiulli and Clara Pizzuti

ICAR-CNR, Via P. Bucci 41C, Università della Calabria, 87036 Rende (CS), Italy  
{angiulli, pizzuti}@icar.cnr.it

**Abstract.** A biclustering algorithm, based on a greedy technique and enriched with a local search strategy to escape poor local minima, is proposed. The algorithm starts with an initial random solution and searches for a locally optimal solution by successive transformations that improve a gain function, combining the mean squared residue, the row variance, and the size of the bicluster. Different strategies to escape local minima are introduced and compared. Experimental results on yeast and lymphoma microarray data sets show that the method is able to find significant biclusters.

## 1 Introduction

In the past recent years, DNA *microarray* technology has captured the attention of scientific community because of its capability of simultaneously measure the activity and interactions of thousands of genes. The relative abundance of the mRNA of a gene under a specific experimental condition (or sample) is called the *expression level* of a gene. The expression level of a large number of genes of an organism under various experimental conditions can be arranged in a data matrix, also known as *gene expression data matrix*, where rows correspond to genes and columns to conditions. Thus each entry of this matrix is a real number representing the expression level of a gene under a specific experiment. One of the objectives of gene expression data analysis is to group genes according to their expression under multiple conditions. Clustering [4,11,1] is an important gene expression analysis method that has been extensively used to group either genes, to search for functional similarities, or conditions, to find samples characterized by homogeneous gene expression levels. However, generally, genes are not relevant for all the experimental conditions, but groups of genes are often co-regulated and co-expressed only under specific conditions. This important observation has lead the attention towards the design of clustering methods that try to simultaneously group genes and conditions. The approach, named *biclustering*, detects subsets of genes that show similar patterns under a specific subset of experimental conditions.

Biclustering was first defined by Hartigan [7] and called *direct clustering*. His aim was to find a set of sub-matrices having zero variance, that is with constant values. This concept was then adopted by Cheng and Church [2] by introducing

a similarity score, called *mean squared residue*, to measure the coherence of rows and columns in the bicluster. A group of genes is considered coherent if their expression levels varies simultaneously across a set of conditions. Biclusters with a high similarity score and, thus, with low residue, indicate that genes show similar tendency on the subset of the conditions present in the bicluster. The problem of finding biclusters with low mean squared residue, in particular maximal biclusters with scores under a fixed threshold, has been proved to be NP-hard [2] because it includes the problem of finding a maximum biclique in a bipartite graph as a special case [5]. Therefore, Cheng and Church proposed heuristic algorithms that are able to generate good quality biclusters.

In this paper a greedy search algorithm to find  $k$  biclusters with a fixed degree of overlapping is proposed. The method is enriched with an heuristic to avoid to get trapped at poor local minima. The algorithm starts with an initial random bicluster and searches for a locally optimal solution by successive transformations that improve a *gain* function. The *gain* combines the mean squared residue, the row variance, and the size of the bicluster. In order to escape poor local minima, that is low quality biclusters having negative gain in their neighborhood, random moves with given probability are executed. These moves delete or add a row/column on the base of different strategies introduced in the method. To obtain  $k$  biclusters the algorithm is executed  $k$  times by allowing to control the degree of overlapping among the biclusters. Experimental results on two well known microarray data sets, *yeast cell cycle* and *B-cell lymphoma*, show that the algorithm is able to find significant and coherent biclusters.

The paper is organized as follows. The next section defines the problem of biclustering and the notations used. In Section 3 an overview of the existing approaches to biclustering is given, section 4 describes the algorithm proposed, and, finally, section 5 reports the experiments on the two mentioned data sets.

## 2 Notation and Problem Definition

In this section the notation used in the paper is introduced and a formal definition of bicluster is provided [2]. Let  $X = \{I_1, \dots, I_N\}$  be the set of genes and  $Y = \{J_1, \dots, J_M\}$  be the set of conditions. The data can be viewed as an  $N \times M$  matrix  $A$  of real numbers. Each entry  $a_{ij}$  in  $A$  represents the relative abundance (generally its logarithm) of the mRNA of a gene  $I_i$  under a specific condition  $J_j$ .

A *bicluster* is a sub-matrix  $(I, J)$  of  $A$ , where  $I$  is a subset of the rows  $X$  of  $A$ , and  $J$  is a subset of the columns  $Y$  of  $A$ .

Let  $a_{iJ}$  denote the mean of the  $i$ th row of the bicluster  $(I, J)$ ,  $a_{iJ}$  the mean of the  $j$ th column of  $(I, J)$ , and  $a_{IJ}$  the mean of all the elements in the bicluster. More formally,

$$a_{iJ} = \frac{1}{|J|} \sum_{j \in J} a_{ij}, \quad a_{iJ} = \frac{1}{|I|} \sum_{i \in I} a_{ij}, \quad a_{IJ} = \frac{1}{|I||J|} \sum_{i \in I, j \in J} a_{ij}.$$

The *volume*  $v_{IJ}$  of a bicluster  $(I, J)$  is the number of entries  $a_{ij}$  such that  $i \in I$  and  $j \in J$ , that is  $v_{IJ} = |I| \times |J|$ .

The *residue*  $r_{ij}$  of an element  $a_{ij}$  is defined as  $r_{ij} = a_{ij} - a_{iJ} - a_{IJ} + a_{IJ}$ . The residue of an element provides the difference between the actual value of  $a_{ij}$  and its expected value predicted from its row, column, and bicluster mean. The residue of an element reveals its degree of coherence with the other entries of the bicluster it belongs to. The lower the residue, the higher the coherence. The quality of a bicluster can be thus evaluated by computing *the mean squared residue*  $r_{IJ}$ , i.e. the sum of all the squared residues of its elements:

$$r_{IJ} = \frac{\sum_{i \in I, j \in J} (r_{ij})^2}{v_{IJ}}.$$

The *mean squared residue* of a bicluster, as outlined by Cheng and Church in [2], provides the similarity score of a bicluster. Given a threshold  $\delta \geq 0$ , a sub-matrix  $(I, J)$  is said a  $\delta$ -*bicluster*, if  $r_{IJ} < \delta$ . The aim is then to find large biclusters with scores below a fixed threshold  $\delta$ . However, low residue biclusters should be accompanied with a sufficient variation of the gene values with respect to the row mean value, otherwise trivial biclusters having almost all constant values could be determined. To this end the row variance  $var_{IJ}$  of a bicluster  $(I, J)$  is defined as

$$var_{IJ} = \frac{\sum_{i \in I, j \in J} (a_{ij} - a_{iJ})^2}{v_{IJ}}.$$

The final goal is to obtain large biclusters, with a relatively high variance, and with mean squared residue lower than a given threshold  $\delta$ .

### 3 Related Work

A comprehensive survey on biclustering algorithms for biological data analysis can be found in [9]. In the following the main existing proposals will be described. As already mentioned in the introduction, Hartigan [7] first suggested a partition based algorithm, called *direct clustering*, that splits the data matrix to find sub-matrices having zero variance, that is with constant values. Hartigan used the variance of a bicluster to evaluate its quality and his aim was to obtain constant sub-matrices. However, he proposed to modify his algorithm to find biclusters with coherent values in rows and columns. Cheng and Church [2] were the first who introduced the new paradigm of biclustering to gene expression data analysis. They proposed some greedy search heuristics that generate sub-optimal biclusters satisfying the condition of having the mean squared residue below a threshold  $\delta$ . The heuristics start with the original data matrix and add or delete rows and columns. The algorithms assume that the data matrix doesn't contain missing values and can find one or  $k$  biclusters. In the latter case, in order to avoid to reobtain the same biclusters, the values of those elements  $a_{ij}$  that have already been inserted in a bicluster are substituted with random numbers. Yang et al. [12] extended the definition of  $\delta$ -bicluster to cope with missing values and to avoid problems caused by random numbers. In fact, they experimented that random numbers in the methods of Cheng and Church can

interfere with the discovery of new biclusters, in particular for those that overlap with those already obtained. They defined a probabilistic *move-based* algorithm FLOC (FLexible Overlapped biClustering) that generalizes the concept of mean squared residue and based on the concept of *action* and *gain*. Getz et al. [6] presented the Coupled Two-Way Clustering algorithm that uses a hierarchical clustering method separately on each dimension. Clusters of rows are used as conditions for column clustering and vice-versa. Lazzeroni and Owen [8] introduced the plaid model, where the concept of layers (bicluster) is used to compute the values of the elements in the data matrix. The data matrix is described as a linear function of layers corresponding to its biclusters. Tanay et al. presented SAMBA (Statistical-algorithmic Method for Bicluster Analysis), a biclustering algorithm that combines graph theory and statistics. The data matrix is represented as a bipartite graph where the nodes are conditions and genes, and edges denote significant expression changes. Vertex pairs are associated with a weight, and heavy subgraphs correspond to significant biclusters. Cho et al. [3] propose two iterative co-clustering algorithms that use two similar squared residue measures, and based on the  $k$ -means clustering method. They formulate the problem of minimizing the residue as trace optimization problems that provide a spectral relaxation, used to initialize their methods.

## 4 Algorithm Description

In this section we present *RandomWalkBiclustering*, a biclustering algorithm based on a greedy technique enriched with a local search strategy to escape poor local minima. The basic schema of our method derives from the WSAT algorithm of Selman et al. for the *Satisfiability problem* [10], opportunely modified to deal with the biclustering problem. The algorithm starts with an initial random bicluster  $B = (I, J)$  and searches for a  $\delta$ -bicluster by successive transformations of  $B$ , until a gain function is improved. The transformations consist in the change of membership (called flip or move) of the row/column that leads to the largest increase of the gain function. The *gain* function combines mean squared residue, row variance, and size of the bicluster by means of user-provided weights  $w_{res}$ ,  $w_{var}$ , and  $w_{vol}$ . More formally, let

$$\Delta res = \frac{res_{old} - res_{new}}{res_{old}}, \quad \Delta var = \frac{var_{old} - var_{new}}{var_{old}}, \quad \Delta vol = \frac{vol_{old} - vol_{new}}{vol_{old}},$$

be the relative changes of residue, row variance, and volume when a row/column is added/removed, where  $res_{old}$ ,  $var_{old}$ , and  $vol_{old}$  (resp.  $res_{new}$ ,  $var_{new}$ , and  $vol_{new}$ ) are respectively the values of the residue, row variance and volume of  $B$  before (after) the move. Then the function *gain* is defined as

$$gain = w_{res}(2^{\Delta res} - 1) - w_{var}(2^{\Delta var} - 1) - w_{vol}(2^{\Delta vol} - 1),$$

with  $w_{res} + w_{var} + w_{vol} = 1$ . This function assumes values in the interval  $[-1, +1]$ . In fact, relative changes  $\Delta res$ ,  $\Delta var$ , and  $\Delta vol$  range in the interval  $[-\infty, +1]$ ,

**Algorithm** RandomWalkBiclustering**Input:**

- *matrix*: a gene-expression matrix
- $\delta$ : stop when this value of residue is reached (0=stop at local minimum)
- *max\_flips*: maximum number of iterations allowed
- *method*: type of random move
- *p*: probability of a random move (0 = no random move)
- $w_{res}, w_{var}, w_{vol}$ : weight associated to the residue, row variance, and volume resp.
- $row_{min}, row_{max}$ : minimum and maximum number of rows allowed in the bicluster
- $col_{min}, col_{max}$ : minimum and maximum number of columns allowed in the bicluster

**Method:**

```

generate at random a bicluster that does not violate the constraints on the number
of rows and columns
set  $flips = 0, res = +\infty, local\_minimum = false$ 
while  $flips < max\_flips$  and  $\delta < res$  and not  $local\_minimum$ 
     $flips = flips + 1$ 
    if a random generated number is less than  $p$  then
        execute a random move according to the method chosen, that does not
        violate the constraints on the number of rows and columns, and update
        the residue value  $res$ 
    else
        let  $m$  be the move, that does not violate the constraints on the number
        of rows and columns, with the maximum gain  $gain$ 
        if  $gain > 0$  then
            execute the move  $m$  and update the residue value  $res$ 
        else
            set  $local\_minimum = true$ 
    return the bicluster computed

```

**Fig. 1.** The *RandomWalkBiclustering* algorithm

consequently the terms  $2^{\Delta_{res}}$ ,  $2^{\Delta_{var}}$ , and  $2^{\Delta_{vol}}$  range in the interval  $[0, 2]$ , and the whole function is assumed values between  $-1$  and  $+1$ . The weights  $w_{res}$ ,  $w_{var}$ , and  $w_{vol}$  provide a trade-off among the relative changes of residue, row variance, and volume. When  $w_{res} = 1$  (and thus  $w_{var} = w_{vol} = 0$ ), the algorithm searches for a *minimum residue bicluster*, since the gain monotonically increases with the residue of the bicluster. Decreasing  $w_{res}$  and increasing  $w_{var}$  and  $w_{vol}$ , biclusters with higher row variance and larger volume can be obtained. Notice that when the residue after a flip diminishes, and the row variance and volume increase,  $\Delta_{res}$  is positive, while  $\Delta_{var}$  and  $\Delta_{vol}$  are negative. Thus, when the gain function is positive, *RandomWalkBiclustering* is biased towards large biclusters with a relatively high variance, and low residue. A negative gain, on the contrary, means a deterioration of the bicluster because there could have been either an augmentation of the residue or a decrease of the row variance or volume. During its execution, in order to avoid get trapped into poor local minima (i.e. low quality biclusters with negative gain in their neighborhood), instead of performing the flip maximizing the gain, with a user-provided probability  $p$  the

algorithm is allowed to execute a random move. We introduced three types of random moves:

- NOISE: with probability  $p$ , choose at random a row/column of the matrix and add/remove it to/from  $B$ ;
- REMOVE: with probability  $p$ , choose at random a row/column of  $B$  and remove it from  $B$ ;
- REMOVE-MAX: with probability  $p$ , select the row/column of  $B$  scoring the maximum value of residue, and remove it from  $B$ .

Thus, the NOISE is a purely random strategy that picks a row/column from the overall matrix, and not only from the bicluster, and adds or removes the row/column to the bicluster if it belongs or it does not belong to it. The REMOVE strategy removes at random a row/column already present in the bicluster, thus it could accidentally delete a worthless gene/condition from the current solution, and the REMOVE-MAX removes that row/column already present in the bicluster having the highest value of the residue, i.e. mostly contributing to worsen the gain. Figure 3 shows the algorithm *RandomWalkBiclustering*. The algorithm receives in input a gene-expression matrix, a threshold value ( $\delta$ ) for the residue of the bicluster, the maximum number of times (*max\_flips*) that a flip can be done, the kind of random move the algorithm can choose (*method*), the probability ( $p$ ) of executing a random move, the weight to assign to residue ( $w_{res}$ ), variance ( $w_{var}$ ), and volume ( $w_{vol}$ ), and some optional constraint ( $row_{min}$ ,  $row_{max}$ ,  $col_{min}$ ,  $col_{max}$ ) on the size of the bicluster to find. The flips are repeated until either a preset of maximum number of flips (*max\_flips*) is reached, or a  $\delta$ -bicluster is found, or the solution can not ulteriorly be improved (get trapped into a local minimum). Until the stop condition is not reached, it executes a random move with probability  $p$ , and a greedy move with probability  $(1 - p)$ . In order to compute  $k$  biclusters, we execute  $k$  times the algorithm *Random-WalkBiclustering* by fixing two frequency thresholds,  $f_{row}$  and  $f_{col}$ , that allow to control the degree of overlapping among the biclusters. The former binds a generic row to participate to at most  $k \cdot f_{row}$  biclusters among the  $k$  to be found. Analogously,  $f_{col}$  limits the presence of a column in at most  $k \cdot f_{col}$  biclusters. During the  $k$  executions of the algorithm, whenever a row/column exceeds the corresponding frequency threshold, it is removed from the matrix and not taken into account any more in the subsequent executions.

**Computational Complexity.** The temporal cost of the algorithm is upper bounded by

$$max\_flips \times C_u \times [(1 - p) \times (N + M) + p]$$

where  $C_u$  is the cost of computing the new residue and the new row variance of the bicluster after performing a move. In order to reduce the complexity of  $C_u$ , we maintain, together with the current bicluster  $B = (I, J)$ , the mean values  $a_{i,J}$  and  $a_{I,j}$ , for each  $i \in I$ , the summation  $\sum_{j \in J} a_{ij}^2$ , and the total sum of the row variances. The computation of the new residue of each element involves recomputing the  $|I| + |J|$  mean values  $a_{i,J}$  ( $1 \leq i \leq |I|$ ) and  $a_{I,j}$  ( $1 \leq j \leq |J|$ ) after performing the move. This can be done efficiently, in time  $\max\{|I|, |J|\}$ , by



exploiting the values maintained together with the current bicluster. Computing the residue  $res_{new}$  of the new bicluster, requires the computation of the squares of its element residues, a time proportional to the volume of the new bicluster. We note that, usually,  $|I||J| \ll NM$ . Computing the new row variances can be done in a fast way by exploiting the summations  $\sum_{j \in J} a_{ij}^2$  already stored. Indeed, if a column is added or removed, the new row variances can be obtained quickly by evaluating the  $|I|$  expressions  $\frac{1}{|J|} \sum_{i,j} (a_{ij}^2) - a_{i,J}^2$  ( $1 \leq i \leq |I|$ ). For example, if the  $q$ th column is added, in order to compute the new variance of the row  $i$ , the following expression must be evaluated:

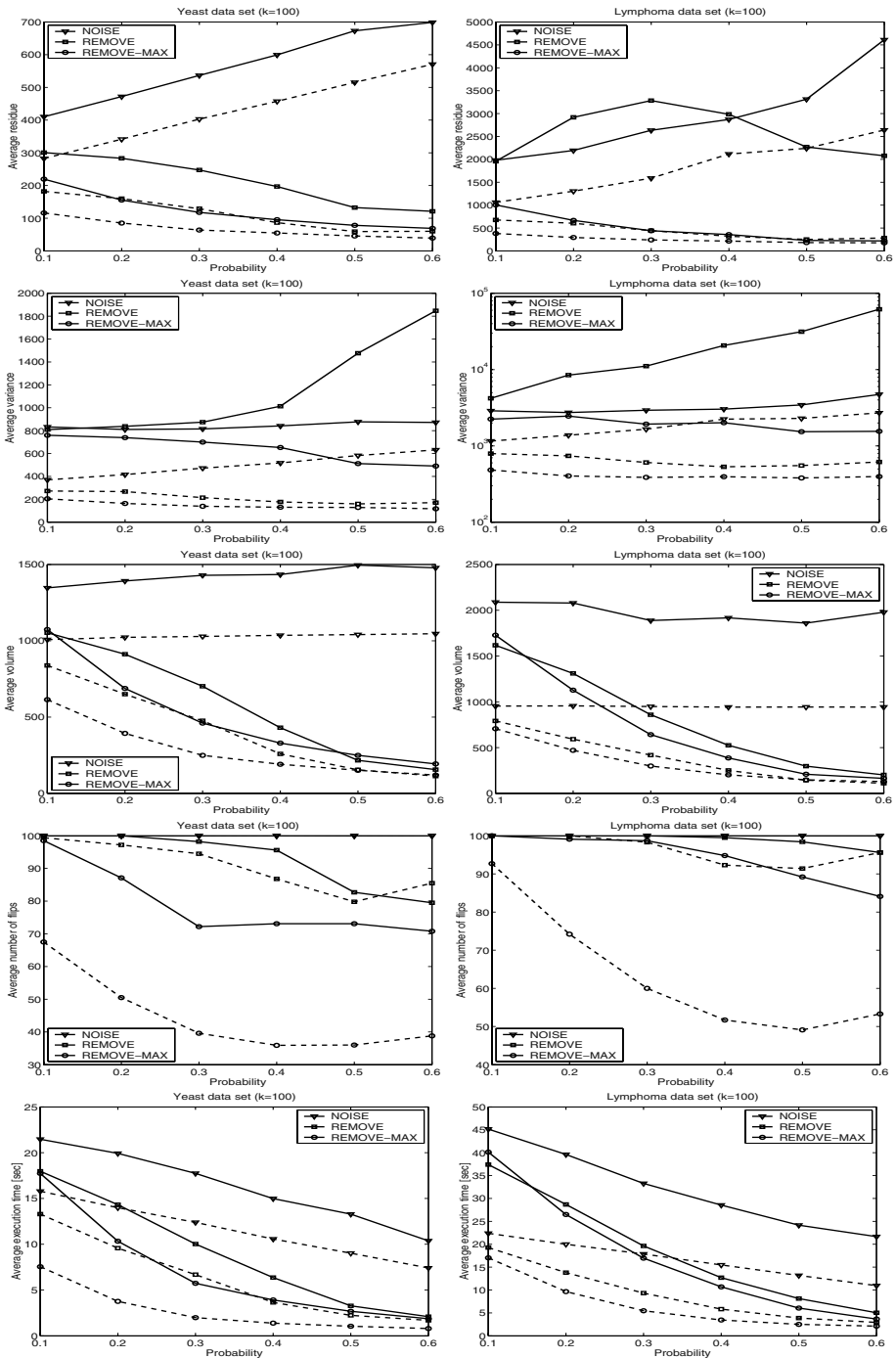
$$\frac{1}{|J|+1} \left( \sum_{j \in J} (a_{ij}^2) + a_{iq}^2 \right) - \left( \frac{|J|a_{iJ} + a_{iq}}{|J|+1} \right)^2.$$

Analogously if a column is removed. Otherwise, if a row is added (removed resp.) the corresponding row variance must be computed and added (subtracted resp.) to the overall sum of row variances. Before concluding, we note that the cost of a random move is negligible, as it consists in generating a random number, when the NOISE or REMOVE strategies are selected, while the row/column with the maximum residue, selected by the REMOVE-MAX strategy, is computed, with no additional time requirements, during the update of the residue of the bicluster at the end of each iteration, and, hence, it is always immediately available.

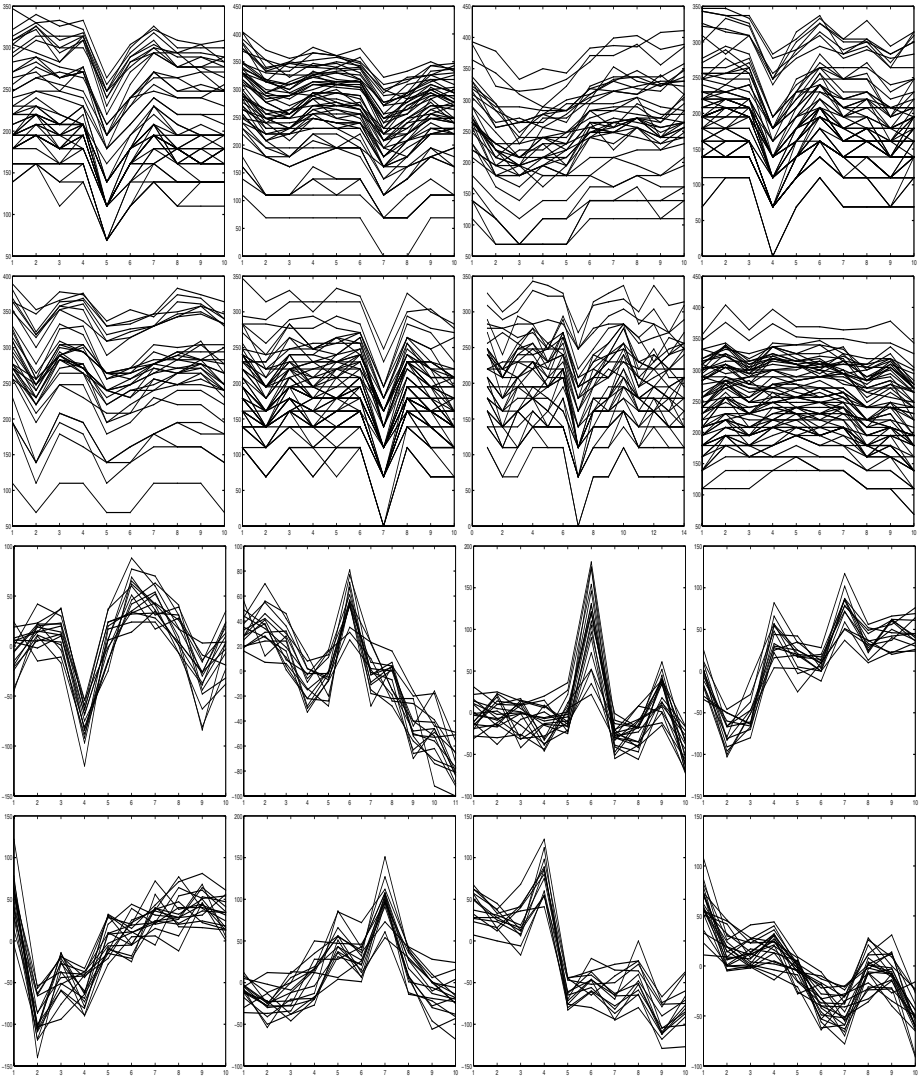
## 5 Experimental Results

In this section we give experimental results to show the behavior of the *RandomWalkBiclustering* algorithm. We selected two well known gene expression data sets, the *Yeast Saccharomyces cerevisiae* cell cycle expression data set, and the human B-cell *Lymphoma* data set. The preprocessed gene expression matrices can be obtained from [2] at <http://arep.med.harvard.edu/biclustering>. The yeast cell cycle data set contains 2884 genes and 17 conditions. The human lymphoma data set has 4026 genes and 96 conditions. The algorithm has been implemented in C, and all the experiments have been performed on a Pentium Mobile 1700MHz based machine. The experiments aimed at comparing the three random move strategies when different probabilities and input parameters are given and to discuss the advantages of each of them. In particular, we computed  $k = 100$  biclusters varying the probability  $p$  of a random move in the interval  $[0.1, 0.6]$ , for two different configurations of the weights, i.e.  $\mathbf{w}_1 = (w_{res}, w_{var}, w_{vol}) = (1, 0, 0)$  (dashed lines in Figure 2) and  $\mathbf{w}_2 = (w_{res}, w_{var}, w_{vol}) = (0.5, 0.3, 0.2)$  (solid lines in Figure 2). Notice that  $w_{res} = 1$  and  $w_{var} = w_{vol} = 0$ , means that the gain function is completely determined by the residue value. We set *max\_flips* to 100,  $\delta$  to 0, and the frequency thresholds to  $f_{row} = 10\%$  and  $f_{col} = 100\%$ , i.e. a row can participate in at most 10 biclusters, while a column can appear in all the 100 biclusters. The initial random generated biclusters are of size  $14 \times 14$  for the Yeast data set and of size  $20 \times 20$  for the Lymphoma data set, while we constrained biclusters to have at least  $row_{min} = 10$  rows and  $col_{min} = 10$  columns. Figure 2 shows the behavior

of the algorithm on the two above mentioned data sets. From the top to the bottom, the figures show the average residue, row variance and volume of the 100 biclusters computed, the average number of flips performed by the method, and the average execution time. Figures on the left concern the Yeast data set, and figures on the right the Lymphoma data set. We can observe that, as regards the residue, the REMOVE-MAX method performs better than the two others, as expected. In fact, its random move consists in removing the gene/condition having the highest residue. Furthermore, increasing the random move probability  $p$  improves the value of the residue. The residue of the NOISE method, instead, deteriorates when the probability increases. The REMOVE strategy, on the Yeast data set is better than the NOISE one, but worse than the REMOVE-MAX. On the Lymphoma data set, the value of the residue increases until  $p = 0.3$  but then it decreases. The residue scored for parameters  $\mathbf{w}_1$  (dashed lines) is lower with respect to that obtained for  $\mathbf{w}_2$  (solid lines), for the two strategies NOISE and REMOVE, while, for REMOVE-MAX the difference is negligible. As regards the variance, we can note that the variance of REMOVE is greater than that of NOISE, and that of NOISE is greater than that of REMOVE-MAX for both  $\mathbf{w}_1$  e  $\mathbf{w}_2$ . This is of course expected, since in the former case we do not consider the variance in the gain function in order to obtain the biclusters, while in the latter the weight of the variance is almost as important as that of the residue (0.3 w.r.t 0.5). Analogous considerations hold for the volume, whose value is higher for  $\mathbf{w}_2$ . Furthermore, the volume is almost constant for the NOISE strategy, because the probability of adding or removing an element in the bicluster is more or less the same, but it decreases for the REMOVE and REMOVE-MAX strategies. These two strategies tend to discovery biclusters having the same size when the probability  $p$  increases. As for the average number of flips, we can note that 100 flips are never sufficient for the NOISE method to reach a local minimum, while the other two methods do not execute all the 100 flips. In particular, the RANDOM-MAX strategy is the fastest since it is that which needs less flips before stopping. As regards the execution time, the algorithm is faster for  $\mathbf{w}_1$  w.r.t  $\mathbf{w}_2$ , but, in general, the execution time decreases when the probability  $p$  increases and they are almost the same for higher values of  $p$  because the number of random moves augments for both. Finally some consideration on the quality of the biclusters obtained. We noticed that the NOISE strategy, which works in a purely random way, gives biclusters with lower gain and it requires more execution time. On the contrary, REMOVE-MAX is positively biased by the removal of those elements in the bicluster having the worst residue, thus it is able to obtain biclusters with higher values of residue and volume, while the REMOVE strategy extract biclusters with higher variance. To show the quality of the biclusters found by *RandomWalkBiclustering*, Figure 3 depicts some of the biclusters discovered in the experiments of Figure 2 by using the REMOVE-MAX strategy for  $(w_{res}, w_{var}, w_{vol}) = (0.5, 0.3, 0.2)$ . The  $x$  axis corresponds to conditions, while the  $y$  axis gives the gene expression level. The figures point out the good quality of the biclusters obtained. In fact, their expression levels vary homogeneously under a subset of conditions, thus they present a high degree of coherence.



**Fig. 2.** Average residue, variance, volume, number of flips and execution time for Yeast (on the left) and Lymphoma (on the right) data sets



**Fig. 3.** Biclusters obtained by using the REMOVE-MAX strategy with  $(w_{res}, w_{var}, w_{vol}) = (0.5, 0.3, 0.2)$  in the experiments of Figure 2. The first two rows show 8 biclusters of the Yeast data set ( $p = 0.3$ ), while the subsequent two rows show 8 biclusters of the Lymphoma data set ( $p = 0.5$ ). From left to right and from top to bottom the values of (residue, variance, volume) are the following: (70.14, 590.65, 460), (99.51, 705.58, 530), (160.89, 834.79, 360), (113.47, 674.25, 630), (83.04, 439.81, 310), (136.31, 788.27, 580), (180.03, 545.23, 518), and (111.01, 356.24, 640) for the Yeast data set, and (214.63, 1414.45, 150), (169.96, 1626.74, 165), (366.18, 2012.5, 190), (181.34, 2135.5, 140), (323.17, 2472.65, 170), (182.45, 1499.95, 160), (200.24, 3412.19, 130), and (172.94, 1197.03, 220) for the Lymphoma data set.

## 6 Conclusions

The paper presented a greedy search algorithm to find overlapped biclusters enriched with a local search strategy to escape poor local minima. The proposed algorithm is guided by a gain function that combines the mean squared residue, the row variance, and the size of the bicluster through user-provided weights. Different strategies to escape local minima have been introduced and compared. Experimental results showed that the algorithm is able to obtain groups of genes co-regulated and co-expressed under specific conditions. Future work will investigate the behavior of the algorithm for many different combinations of the input parameters, in particular for the weights  $w_{res}$ ,  $w_{var}$ , and  $w_{vol}$  employed in the gain function, to study how the trade-off among residue, variance, and volume affects the quality of the solution. We are also planning an extensive comparison with other approaches, and an analysis of the biological significance of the biclusters obtained.

## References

1. A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999.
2. Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference On Intelligent Systems for Molecular Biology (ISMB'00)*, pages 93–103, 2000.
3. H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the the Fourth SIAM International Conference on Data Mining (SDM'04)*, 2004.
4. M. B. Eisen, P. Spellman, P. O. Brown, and P. Botstein. Cluster analysis and display of genome-wide expression pattern. In *Proceedings of the National Academy of Sciences, USA 8*, pages 14863–14868, 1998.
5. M. R. Garey and D.S. Johnson. *Computers and intractability: A guide to the Theory of NP-Completeness*. San Francisco: Freeman, 1979.
6. G. Getz, E. Levine, and E. Domany. Coupled two-way cluster analysis of gene microarray data. In *Proceedings of the National Academy of Sciences, USA*, pages 12079–12084, 2000.
7. J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.
8. L. Lazzeroni and A. Owen. Plaid models for gene expression data. Technical report, Stanford University, 2000.
9. S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.
10. B. Selman, H.A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the 12th Nation Conference on Artificial Intelligence (AAAI'94)*, pages 337–343, 1994.
11. S. Tavazoie, J.D. Hughes, M. Campbell, R.J. Cho, and G.M. Church. Systematic determination of genetic network architecture. *Natural genetics*, 22:281–285, 1999.
12. J. Yang, W. Wang, H. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proceedings of the 3rd IEEE Conference On Bioinformatics and Bioengineering (BIBE'03)*, pages 321–327, 2003.

# Spectral Kernels for Classification

Wenyuan Li<sup>1</sup>, Kok-Leong Ong<sup>2</sup>, Wee-Keong Ng<sup>1</sup>, and Aixin Sun<sup>3</sup>

<sup>1</sup> Nanyang Technological University, Centre for Advanced Information Systems,  
Nanyang Avenue, N4-B3C-14, Singapore 639798

liwy@pmail.ntu.edu.sg, awkng@ntu.edu.sg

<sup>2</sup> School of Information Technology, Deakin University,  
Waurm Ponds, VIC 3217, Australia

leong@deakin.edu.au

<sup>3</sup> School of Computer Science and Engineering, University of New South Wales,  
Sydney, NSW 2052, Australia

aixinsun@cse.unsw.edu.au

**Abstract.** Spectral methods, as an unsupervised technique, have been used with success in data mining such as LSI in information retrieval, HITS and PageRank in Web search engines, and spectral clustering in machine learning. The essence of success in these applications is the spectral information that captures the semantics inherent in the large amount of data required during unsupervised learning. In this paper, we ask if spectral methods can also be used in supervised learning, e.g., classification. In an attempt to answer this question, our research reveals a novel kernel in which spectral clustering information can be easily exploited and extended to new incoming data during classification tasks. From our experimental results, the proposed Spectral Kernel has proved to speedup classification tasks without compromising accuracy.

## 1 Introduction

Kernel-based learning first appear in the form of Support Vector Machines (SVM), and readily became the state-of-the-art for learning algorithms. The framework of kernel-based learning methods (KM) is also known as kernel-based analysis of data in both supervised and unsupervised learning [1,2,3]. Within this framework, kernels encode all the information required by the learning machinery, and acts as the interface between the data and the learning modules [4]. Hence, they are implicitly high-dimensional spaces that contain more information than the original explicit feature space. The advantage of this is that once obtained, kernel algorithms can perform analysis without further information from the original data set.

There have been many success stories [5,6,7,8] with kernels. In text categorization, the kernel was used to capture a semantic network of terms to better compute the similarities between documents [9]. In natural language learning, subparse trees are taken into consideration in the semantic kernel to improve the accuracy of classifying predicative arguments [10]. And in image retrieval, the knowledge about users' queries are encoded in the kernel to improve query accuracy [11]. While domain knowledge is usually encoded in the kernel by the expert user, they can also be obtained from automated discovery algorithms. The pioneering attempt to integrate unsupervised discovery, in

the form of kernels, for supervised learning is Latent Semantic Indexing (LSI). It has been shown [12] that the Latent Semantic Kernel (LSK) benefits from the automated discovery of latent semantics that aid the task of classification. In fact, the semantics uncovered in LSI is simply the ‘tip of the iceberg’ of spectral graph analysis on kernel matrices. Under spectral graph theory, there have been active research on the use of latent semantics for clustering. This research, known as *spectral clustering*, is a method that uses spectral information to assist clustering algorithms.

Obtaining the spectral information of a data set is a three step process: (i) compute the similarity matrix  $\mathbf{S}$  from the data; (ii) transform  $\mathbf{S}$  to another matrix  $\Gamma(\mathbf{S})$ ; and finally (iii) perform an eigen-decomposition on  $\Gamma(\mathbf{S})$ . Our analysis of this process led to an important discovery — if certain matrix transformation (e.g., normalized Laplacian) is performed, we can observe some interesting latent clustering semantics in the eigenvalues and eigenvectors of  $\Gamma(\mathbf{S})$  [5,13,14,15,16,17] that can be used in the kernel for the task of classification. Our observation, and hence the main contribution of this paper, led to our proposal of the *Spectral Kernel*. The spectral kernel combines two state-of-the-art learning algorithms: kernel-based learning and spectral clustering; and introduces a mechanism that supports the spectral embedding of new input data into the kernel to improve classification precision.

We present our proposal as follows. The next section provides the background about kernels and spectral clustering. In doing so, we provide the theoretical analysis and examples to demonstrate the steps to compute the spectral embedding space. We then present in Section 3, the steps to update the kernel values as new input arrives — a differentiation of our approach from other spectral learning methods. We then provide empirical results in Section 4 to support the feasibility of our proposal. Finally, we conclude with related and future work in Section 5.

## 2 Spectral Graph Analysis of Kernel Matrices

To facilitate understanding of our proposal, as well as the analysis and proofs presented in the later sections of this paper, we first introduce some basic facts of kernel matrices and its mathematical foundation [13].

Given a set of data points  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  and a kernel function  $\kappa(\cdot, \cdot)$ , the kernel matrix  $\mathbf{K} = (\mathbf{K}_{ij})_{i,j=1}^n$  is defined as  $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ , where  $\mathbf{K}$  is symmetric and usually positive semi-definite. By operating on  $\mathbf{K}$ , we can easily recode the data in a manner suitable for the learning module. A simple and widely used  $\kappa$  is the inner product  $\kappa_I(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ . And if we have  $\kappa_1(\mathbf{x}, \mathbf{z})$  as a kernel, we can construct new kernels using other kernel functions, e.g., exponential kernel  $\kappa_E(\mathbf{x}, \mathbf{z}) = \exp(\kappa_1(\mathbf{x}, \mathbf{z}))$ ; polynomial kernel  $\kappa_P(\mathbf{x}, \mathbf{z}) = (\kappa_1(\mathbf{x}, \mathbf{z}) + d)^p$  with positive coefficients; and Gaussian kernel  $\kappa_G(\mathbf{x}, \mathbf{z}) = \exp((\kappa_1(\mathbf{x}, \mathbf{x}) + \kappa_1(\mathbf{z}, \mathbf{z}) - 2\kappa_1(\mathbf{x}, \mathbf{z})) / (2\sigma^2))$ .

In many cases,  $\kappa$  need not be an explicit function if the kernel matrix can be given directly. Examples of that include the Latent Semantic Kernel and our proposed Spectral Kernel. The idea is that while some kernels can be represented using explicit functions, many are implicitly represented without one. Regardless of whether  $\kappa$  is an explicit function, the matrix serves as the underlying representation of a kernel capturing all the information required for supervised or unsupervised learning. More interesting perhaps,

**Table 1.** Solution to two graph cut criteria used in spectral clustering: (i) the corresponding transformation by eigen-decomposition of the matrix  $\Gamma(\mathbf{S})$ ; (ii) on an incoming data  $\mathbf{x}$ , the similarity with the training examples is computed as  $\mathbf{S}_x$  and its corresponding transformation is  $\tau(\mathbf{S}_x)$ . *Note:* the original version of the normalized Laplacian matrix should be  $\Gamma_N(\mathbf{S}) = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{S})\mathbf{D}^{-1/2}$ ; see Section 2.2 and Lemma 1.

		Average Volume	Normalized Cut
	Criterion	$\max \frac{\text{vol}(A)}{ A } + \frac{\text{vol}(B)}{ B }$	$\min \frac{\text{cut}(A,B)}{\text{vol}(A)} + \frac{\text{cut}(A,B)}{\text{vol}(B)}$
	Solution to criterion	$\mathbf{S}\mathbf{x} = \lambda\mathbf{x}$	$(\mathbf{D} - \mathbf{S})\mathbf{x} = \lambda\mathbf{D}\mathbf{x}$
(i)	Transformed $\mathbf{S}$	$\Gamma_I(\mathbf{S}) = \mathbf{S}$	$\Gamma_N(\mathbf{S}) = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$
(ii)	Transformed $\mathbf{S}_x$	$\tau_I(\mathbf{S}_x) = \mathbf{S}_x$	$\tau_N(\mathbf{S}_x) = \mathbf{D}^{-1/2}\mathbf{S}_x\mathbf{d}_x^{-1/2}$

is that the underlying idea found in spectral embedding and clustering methods (proposed in recent years) coincides with that of kernel-based methods, i.e., a symmetric matrix is used in the analysis. As a result, there are some interesting properties that we can learn about kernels through spectral properties.

A symmetric matrix  $\mathbf{S} = (\mathbf{S}_{ij})_{n \times n}$  (where  $\mathbf{S}_{ij} = \mathbf{S}_{ji}$ ) is naturally mapped to an undirected graph  $\mathcal{G}(\mathbf{S})$ , where its adjacency matrix is  $\mathbf{S}$ . In spectral graph theory, the spectral component of the transformed  $\mathbf{S}$  has a natural relationship with the structure and properties of the graph  $\mathcal{G}(\mathbf{S})$  [13]. Further let  $\mathcal{G}(\mathbf{S}) = \langle V, E, \mathbf{S} \rangle$  be the graph of  $\mathbf{S}$ , where  $V$  is the set of  $n$  vertices and  $E$  is the set of weighted edges. Each vertex  $i$  of  $\mathcal{G}(\mathbf{S})$  corresponds to the  $i$ -th column (or row) of  $\mathbf{S}$ , and the weight of each edge  $\hat{i}\hat{j}$  corresponds to the non-diagonal entry  $\mathbf{S}_{ij}$ . For any two vertices  $(i, j)$ , a larger value of  $\mathbf{S}_{ij}$  indicates a higher connectivity, and vice versa. From the above, we have the following interesting spectral properties:

**Eigenvalues.** The spectrum of the Normalized Laplacian transformation of  $\mathbf{S}$  reveals the embedding clustering structure of  $\mathcal{G}(\mathbf{S})$  with different global bisection (or cut) criteria [5,13].

**Eigenvectors.** Correspondingly, the  $i$ -th eigenvector naturally explains the meaning of the  $i$ -th eigenvalue. This led to the development of spectral clustering [15,16,17].

## 2.1 Graph Cut Criteria of Kernel Matrices

In spectral clustering, since  $\mathbf{S}$  is actually an adjacency matrix of the weighted graph  $\mathcal{G}(\mathbf{S})$ , finding the clustering structure of  $\mathbf{S}$  can be transformed into the problem of finding an optimum graph cut in  $\mathcal{G}(\mathbf{S})$ . Notably, a different graph cut criterion leads to a different solution of  $\mathcal{G}(\mathbf{S})$ .

In the case of Table 1, the criterion is to find an optimal cut of  $\mathcal{G}(\mathbf{S})$  such that we have two non-overlapping subsets  $A, B \subseteq V$  satisfying the conditions  $A \cap B = \emptyset$  and  $A \cup B = V$ , where  $|A|$  is the number of vertices or data points;  $\text{vol}(A) = \sum_{i \in A} d_i$  is the volume with  $d_i = \sum_{j \in V} \mathbf{S}_{ij}$  being the degree of the vertex  $i$ ;  $\text{cut}(A, B) = \sum_{i \in A, j \in B} \mathbf{S}_{ij}$  is the cut between  $A$  and  $B$ ; and  $\mathbf{D}$  is the diagonal matrix formed from the degrees of the vertices. In both solutions, the second largest eigenvalues (a.k.a. interested eigenvalues) and the corresponding eigenvectors relating to the equations in Table 1 provide the global optimum.



Clearly, the idea of bisecting the kernel matrix can be extended to the first  $k$  largest eigenvalues or eigenvectors. And this observation leads to the construction of multi-way spectral clustering. In the criterion of *average volume*, if we consider the inner product matrix of the term-document matrix (without normalization) as  $\mathbf{S}$ , then its first  $k$  largest eigenvectors is used in LSI for information retrieval. In the criterion of *normalized cut*, the second largest eigenvector of  $\Gamma_N(\mathbf{S})$  is also the clustering information used in the normalized cut image segmentation algorithm [15]. Finally, we have the NJW clustering algorithm [17] when we consider the first  $k$  largest eigenvectors of  $\Gamma_N(\mathbf{S})$ .

We can thus conclude the following. First, the criterion determines the solution and transformation of the kernel matrix that in turn, affects the behavior in the learning module. Second, the type of application to be delivered by the kernel is determined by how the eigenvalues and/or eigenvectors are used. The spectral kernel, presented next, is the result of exploiting these observations.

## 2.2 Computing the Spectral Embedding Space

Among the different spectral clustering techniques proposed in the literature, e.g., [15,16,17], an analysis of the underlying mathematical representation suggests that with appropriate transformations, they essentially reduce to a common representation. This observation motivates the first contribution of the spectral kernel – a unifying framework which by means of different parameters, creates different kernel instances that exhibit different behaviors. We will first prove the existence of this framework, and then show how the spectral embedding is computed.

From Table 1, we see that it is easy to mathematically transform the solutions into a standard eigendecomposition problem of symmetric matrices, i.e.,  $\Gamma(\mathbf{S})\mathbf{x} = \lambda\mathbf{x}$ . To do so however, requires the transformation of  $\mathbf{S}$ , and this is dependent on the solution to the cut criteria. In Table 1,  $\Gamma_I(\mathbf{S})$  represents the original matrix while  $\Gamma_N(\mathbf{S})$  is the normalized Laplacian matrix. From spectral graph theory [13],  $\mathbf{K}_1 = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{S})\mathbf{D}^{-1/2}$  is actually the normalized Laplacian matrix. Notably,  $\mathbf{K}_1$  has the same eigenvectors as  $\mathbf{K}_2 = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$  and the eigenvalues is related by  $\text{eig}(\mathbf{K}_1) = \{1 - \lambda | \lambda \in \text{eig}(\mathbf{K}_2)\}$ , where  $\text{eig}(\cdot)$  is the set of eigenvalues of a symmetric matrix. Furthermore, the interested eigenvalues change from the smallest in  $\mathbf{K}_1$  to the largest in  $\mathbf{K}_2$ . Therefore, it is actually possible to compute the normalized Laplacian matrix using  $\mathbf{K}_2$  giving us  $\Gamma_N(\mathbf{S}) = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$ . In fact, the equivalence relationship between  $\mathbf{K}_1$ ,  $\mathbf{K}_2$ , and the stochastic matrix  $\mathbf{P} = \mathbf{D}^{-1}\mathbf{S}$  can be proven, and therefore in the remaining part of this paper, we will use  $\mathbf{K}_2$  in place of  $\mathbf{K}_1$  and  $\mathbf{P}$  if any.

**Lemma 1 (Equivalence of  $\mathbf{K}_1$ ,  $\mathbf{K}_2$  and  $\mathbf{P}$ ).** *If  $\lambda$  and  $\mathbf{x}$  are correspondingly the eigenvalue and eigenvector of matrix  $\mathbf{K}_1$ , then  $(1 - \lambda)$  are the eigenvalues of the matrices  $\mathbf{K}_2$  and  $\mathbf{P}$ ; and the eigenvectors of  $\mathbf{K}_2$  and  $\mathbf{P}$  are  $\mathbf{x}$  and  $\mathbf{D}^{-1/2}\mathbf{x}$  respectively.*

*Proof.* By definition of  $\mathbf{K}_1$ ,  $\mathbf{K}_2$  and  $\mathbf{P}$ , we have:

$$\mathbf{K}_1 = \mathbf{I} - \mathbf{K}_2 \tag{1}$$

$$\mathbf{K}_2 = \mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2} \tag{2}$$

Suppose  $\lambda$  and  $\mathbf{x}$  are eigenvalue and eigenvector of  $\mathbf{K}_1$ , i.e.,  $\mathbf{K}_1\mathbf{x} = \lambda\mathbf{x}$ . By Equation (1), substituting  $\mathbf{K}_1$  with  $\mathbf{K}_2$  gives us  $(\mathbf{I} - \mathbf{K}_2)\mathbf{x} = \lambda\mathbf{x}$ . After transformation, we

have  $\mathbf{K}_2\mathbf{x} = \mathbf{I}\mathbf{x} - \lambda\mathbf{x} = (1 - \lambda)\mathbf{x}$ . This proves the relationship between  $\mathbf{K}_1$  and  $\mathbf{K}_2$ . We next complete the proof by showing the equivalence of  $\mathbf{K}_2$  and  $\mathbf{P}$ . Suppose now  $\lambda$  and  $\mathbf{x}$  are eigenvalue and eigenvector of  $\mathbf{K}_2$ , i.e.,  $\mathbf{K}_2\mathbf{x} = \lambda\mathbf{x}$ . By Equation (2), substituting  $\mathbf{K}_2$  with  $\mathbf{P}$  gives us  $\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2}\mathbf{x} = \lambda\mathbf{x}$ . By left-multiplication of the matrix  $\mathbf{D}^{-1/2}$  on this equation, we get  $\mathbf{P}\mathbf{D}^{-1/2}\mathbf{x} = \lambda\mathbf{D}^{-1/2}\mathbf{x}$ . Here,  $\lambda$  is the eigenvalue of  $\mathbf{P}$ , and  $\mathbf{D}^{-1/2}\mathbf{x}$  is the eigenvector of  $\mathbf{P}$ .

Within this framework, we can compute the spectral embedding for any specific instance of the spectral kernel. The steps to do so are given in Figure 1. After the spectral components of  $\Gamma(\mathbf{S})$  is computed, the  $k$  interested extreme eigenvalues and eigenvectors are selected to construct the reduced data space. Let the first  $k$  interested eigenvalues of  $\Gamma(\mathbf{S})$  be  $\lambda_1 \triangle \lambda_2 \dots \triangle \lambda_k$ , where “ $\triangle$ ” is “ $\leq$ ” or “ $\geq$ ” according to the different matrix transformation  $\Gamma$ , and  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$  as their corresponding eigenvectors each of dimension  $n$ . The  $k$  dimensional data space is constructed by the two steps shown in Figure 1. The first step has two implementations that can be selected based on the desired application behavior.

Step 1(a) has been proved to be effective and useful on  $\Gamma_N(\mathbf{S})$  in revealing the clustering structure of  $\mathbf{S}$  in [16]. When considered with Step 2, its effectiveness was proven, both theoretically and empirically, in [17]. When Step 1(a) is used with  $\Gamma_I(\mathbf{S})$ , it has proven applications in latent semantic analysis and indexing [6,18]. Step 1(b) on the other hand is well-suited in the context of  $k$ -rank approximation when used with  $\Gamma(\mathbf{S})$  as supported by Lemma 2 below. Furthermore, latent semantic analysis has shown that the  $k$ -rank approximation of a similarity matrix (that is also  $\Gamma_I(\mathbf{S})$ ) incorporates semantic information in measure of similarity between two data points (the same conclusion was also given in latent semantic kernels). We will elaborate this point in Section 3.3.

**Lemma 2 (Approximation of  $\Gamma(\mathbf{S})$  by embedding of Step 1(b)).** *The matrix  $\mathbf{S}'$ , computed by the embedding of Step 1(b) using inner product (i.e.,  $\mathbf{S}'_{ij} = \mathbf{y}_i^T \mathbf{y}_j$ ), is the best  $k$ -rank matrix approximation of  $\Gamma(\mathbf{S})$ .*

*Proof.* Lemma 2 is a variant of the Eckart-Young theorem [19]. Given  $\mathbf{S}_{n \times n} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}$  (singular value decomposition),  $\mathbf{A} = \mathbf{S}_k = \mathbf{U}_k\mathbf{\Lambda}_k\mathbf{V}_k$  is the best rank- $k$  approximation to  $\mathbf{S}$  that minimizes  $\|\mathbf{A} - \mathbf{S}\|_F^2$  among all matrices  $\mathbf{A}$  with rank  $k$  ( $F$  denotes Frobenius norm of a matrix). And because  $\mathbf{S}$  is symmetric, singular values and vectors of  $\mathbf{S}$  are the same as eigenvalues and eigenvectors of  $\mathbf{S}$ .

### 3 Spectral Kernels

In the previous section, the spectral graph analysis of the kernel matrix shows that the spectral embedding (obtained by either Step 1(a) or 1(b)) reveals more latent semantics than the original kernel matrix. By projecting the original feature vectors onto the spectral embedding subspace, we can define a kernel, originating from this subspace, through a particular choice of similarity measure. This effectively registers the clustering information inherent in the subspace into the spectral kernel (SK).

Computing the spectral kernel for classification can be done in three phases: (i) transformation, (ii) spectral embedding, and (iii) kernel computation. Thus, we define

**Step 1(a).**

Directly get  $\mathbf{y}_i = (\mathbf{v}_1^i, \mathbf{v}_2^i, \dots, \mathbf{v}_k^i)^T$ , where  $i=1, 2, \dots, n$ .

**Step 1(b).**

Compute  $\mathbf{y}_i = (\sqrt{\lambda_1} \mathbf{v}_1^i, \sqrt{\lambda_2} \mathbf{v}_2^i, \dots, \sqrt{\lambda_k} \mathbf{v}_k^i)^T$ , where  $i=1, 2, \dots, n$  or  $\mathbf{\Lambda}_k^{1/2} (\mathbf{v}_1^i, \mathbf{v}_2^i, \dots, \mathbf{v}_n^i)^T$ .

**Step 2 (Optional).**

Renormalize each  $\mathbf{y}_i$  to have the unit length (i.e.  $\mathbf{y}_i = \frac{1}{\|\mathbf{y}_i\|} \mathbf{y}_i$ ).

**Fig. 1.** Spectral embedding in the spectral kernel. Let the projected  $k$ -dimensional data space be  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n \in \mathbb{R}^{k \times 1}$ , and the first  $k$  interested eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_k$  are positive. *Note:*  $\mathbf{v}_j^i$  denotes the  $i$ -th coordinate of the eigenvector  $\mathbf{v}_j$ .  $\mathbf{\Lambda}_k$  is the truncated diagonal of  $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ , its last  $(n - k)$  diagonal entries are set 0.

the spectral kernel with three components, i.e.,  $\text{SK}(\mathbf{T}, \mathbf{E}, \mathbf{S})$ , where  $\mathbf{T}$  is the transformation in Table 1;  $\mathbf{E}$  is the embedding step in Figure 1; and  $\mathbf{S}$  is one of the similarity measure in Table 2 selected to compute the final spectral kernel value in the spectral embedding subspace.

In classification, the kernel can contain values from either the training set, or from the training set and its new input (from the testing set). Since during transformation and spectral embedding, the input kernel matrix  $\mathbf{S}$  only holds kernel values from the training set, the spectral embedding can be computed by following the steps given in Section 2.2. In the case where we need to compute the kernel values from both the training and testing set, a different way to compute the spectral embedding of the new input within the same subspace of the training set is needed.

### 3.1 Transforming and Spectral Embedding of New Input

When a new input arrives, its spectral embedding is computed in a similar fashion as described in Section 2.2. The difference is that the transformation and computation of spectral embedding is applied to the vector  $\mathbf{S}_x$  rather than the symmetric matrix. This gives rise to a different transformation and computation of the spectral embedding. After getting the spectral embedding of the new input, the kernel values can be updated with the same similarity measure used during training.

The new input can be given in the form of a vector containing kernel values, i.e.,  $\mathbf{S}_x = (\mathbf{S}_{1x}, \mathbf{S}_{2x}, \dots, \mathbf{S}_{nx})^T$ , where  $\mathbf{S}_{ix}$  represents the kernel value between the  $i$ -th training example and itself. The rationale to why we used  $\mathbf{S}_x$  instead of the vector  $\mathbf{x}$  in the original space is that spectral kernels are based on the other input kernels. In order to compute the spectral embedding of the new input, there is a need to recompute the transformation and the embedding space. Therefore, the vector transformation corresponding to its matrix transformation  $\Gamma(\mathbf{S})$  is defined as  $\tau(\mathbf{S}_x)$  and is given in Table 1 for different  $\Gamma$ . After obtaining  $\tau(\mathbf{S}_x)$ , the following lemma defines how the spectral embedding of the new input is computed.

**Lemma 3 (Spectral embedding of new input).** *Given a kernel matrix  $\mathbf{S}$  for training data, its transformation  $\Gamma(\mathbf{S})$ , the  $k$  interested eigenvalues/vectors of  $\Gamma(\mathbf{S})$  ( $\lambda_i \geq 0$  and  $\mathbf{v}_i, i = 1, 2, \dots, k$ ), and a new input  $\mathbf{S}_x$  in form of kernel values, the spectral embedding of  $\mathbf{S}_x$  for Step 1(a) is  $\mathbf{y} = \mathbf{\Lambda}_k^{-1} \mathbf{V}^T \tau(\mathbf{S}_x)$ , and Step 1(b) is  $\mathbf{y} =$*

**Table 2.** The similarity measures  $s(\mathbf{x}, \mathbf{y})$  used in the computation of spectral kernels

	Inner product	Extension of Euclidean distance	Pearson correlation coefficient
Symbol	$S_I$	$S_E$	$S_P$
Formula	$\mathbf{x}^T \mathbf{y}$	$\exp(-\frac{\ \mathbf{x}-\mathbf{y}\ ^2}{\sigma})$	$\frac{(\mathbf{x}-\bar{\mathbf{x}})^T(\mathbf{y}-\bar{\mathbf{y}})}{\ \mathbf{x}-\bar{\mathbf{x}}\  \ \mathbf{y}-\bar{\mathbf{y}}\ }$

$\Lambda_k^{-1/2} \mathbf{V}^T \tau(\mathbf{S}_x)$ ; where the diagonal matrix  $\Lambda_k = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_k, 0, \dots, 0)$ , and  $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k, \mathbf{v}_{k+1}, \dots, \mathbf{v}_n)^T$ .

*Proof.* By eigendeomposition,  $\Gamma(\mathbf{S}) = \mathbf{V} \Lambda \mathbf{V}^T$ , where  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ . Therefore,  $\Gamma(\mathbf{S})$  can be approximated by  $\Gamma(\mathbf{S}) \approx \mathbf{V} \Lambda_k \mathbf{V}^T = (\Lambda_k^{1/2} \mathbf{V}^T)^T (\Lambda_k^{1/2} \mathbf{V}^T)$ , since the interested  $k$  eigenvalues are the largest positive eigenvalues of  $\Gamma(\mathbf{S})$ . Therefore, each training example  $i$  can be represented by the spectral embedding  $\mathbf{y}_i = \Lambda_k^{1/2} (\mathbf{v}_1^i, \mathbf{v}_2^i, \dots, \mathbf{v}_n^i)^T$  in terms of matrix approximation, where  $\mathbf{y}_i$  is the spectral embedding of the  $i$ -th training example by Step 1(b) from Table 1. If we assume that  $\mathbf{y}$  is also the spectral embedding of the new input in terms of matrix approximation, then  $\mathbf{y}$  should be the spectral embedding obtained by Step 1(b). By matrix approximation, we can therefore approximate the transformed kernel vector  $\tau(\mathbf{S}_x)$  of the new input  $\mathbf{S}_x$  in the same way. Thus, we have  $\tau(\mathbf{S}_x) = (\Lambda_k^{1/2} \mathbf{V}^T)^T \mathbf{y}$ . Since  $\Lambda_k^{1/2} \mathbf{V}^T$  is an orthogonal matrix, it can be solved by taking the matrix inverse to obtain  $\mathbf{y} = \Lambda_k^{-1/2} \mathbf{V}^T \tau(\mathbf{S}_x)$ . This gives the result in Step 1(b). Further, the spectral embedding by Step 1(a) can be computed by multiplying the diagonal matrix  $\Lambda_k^{-1/2}$ , which gives the spectral embedding of the new input  $\mathbf{y} = \Lambda_k^{-1/2} (\Lambda_k^{-1/2} \mathbf{V}^T \tau(\mathbf{S}_x)) = \Lambda_k^{-1} \mathbf{V}^T \tau(\mathbf{S}_x)$  by Step 1(a).

Essentially, Lemma 3 specifies how to project the new input onto the spectral embedding space given by the training examples. And Step 2 of Figure 1 served as the optional step that can be applied to the spectral embedding of the new input (computed either by Step 1(a) or 1(b)), and its used is dependent on whether Step 2 was used during training so that the new input can be compared with the training examples within the same embedding space.

### 3.2 Computing the Spectral Kernel

When the spectral embedding of the training and testing set is ready, the final step is to compute the spectral kernel values from the spectral embedding using a selected similarity measure. This step is flexible and many typical similarity measures can be used. Table 2 lists some possible options for the spectral kernel. Notice that the magnitude of the similarity measure need not be constrained within 0 and 1 since there is no strict requirement on the range of possible kernel values in classification.

### 3.3 Relationship to Latent Semantic Kernel

We conclude this section with the proof that the latent semantic kernel is only a specific instance of the spectral kernel. The objective is two-fold: (i) we want to clarify the

difference between spectral kernels and latent semantic kernels as they appear similar at first glance; and (ii) by this proof, we seek to establish spectral kernels as a framework under which spectral clustering information can be further researched to improve the task of classification.

**Theorem 1.** *Given the term-document matrix  $\mathbf{D}$ , the latent semantic kernel is a specific instance of the spectral kernel, i.e.,  $SK\langle\Gamma_I(\mathbf{D}^T\mathbf{D}), 1(b), S_I\rangle$  reduces to the latent semantic kernel.*

*Proof.* From [12], we have the following facts: latent semantic kernels are computed from the term-document matrix  $\mathbf{D}$  or  $\mathbf{S} = \mathbf{D}^T\mathbf{D}$ ; the LSK matrix of training set is  $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}_k\mathbf{V}^T$ ; the LSK values between the training set  $\mathbf{d}_i$  and the new input  $\mathbf{d}$  is  $\kappa(\mathbf{d}_i, \mathbf{d}) = (\mathbf{V}\mathbf{\Lambda}_k\mathbf{V}^T\mathbf{t})_i$ , where  $\mathbf{t} = \mathbf{D}^T\mathbf{d}$  and the matrix  $\mathbf{V}$  is obtained from eigen decomposition  $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ . Using the above, we prove that spectral kernels with a configuration of  $SK\langle\Gamma_I(\mathbf{S}), 1(b), S_I\rangle$  gives the same  $\mathbf{K}$  and  $\kappa(\mathbf{d}_i, \mathbf{d})$  as LSK. We denote  $\hat{\mathbf{K}}$  as the SK matrix of the training set and  $\hat{\kappa}(\mathbf{d}_i, \mathbf{d})$  as the SK values between the training set  $\mathbf{d}_i$  and the new input  $\mathbf{d}$ . Since the input kernel matrix of SK is  $\mathbf{S} = \mathbf{D}^T\mathbf{D}$  and is computed by the inner product of each document  $\mathbf{d}_i$ , we can easily get the input kernel values of the new input  $\mathbf{d}$  as  $\mathbf{S}_x = \mathbf{D}^T\mathbf{d}$ . As the transformation is  $\Gamma_I$ , we have  $\Gamma_I(\mathbf{S}) = \mathbf{S}$  and  $\tau_I(\mathbf{S}_x) = \mathbf{S}_x = \mathbf{D}^T\mathbf{d} = \mathbf{t}$ . Then, we compute the spectral embedding (using Step 1(b)) of  $\mathbf{d}_i$  as  $\mathbf{y}_i = \mathbf{\Lambda}_k^{1/2}(\mathbf{v}_1^i, \mathbf{v}_2^i, \dots, \mathbf{v}_n^i)^T$ , where  $\mathbf{v}_i$  is the  $i$ -th eigenvector of  $\mathbf{S}$  or the  $i$ -th column of  $\mathbf{V}$ . Further, the spectral embedding of the new input  $\mathbf{d}$  (by Step 1(b)) is  $\mathbf{y} = \mathbf{\Lambda}_k^{-1/2}\mathbf{V}^T\tau(\mathbf{S}_x) = \mathbf{\Lambda}_k^{-1/2}\mathbf{V}^T\mathbf{t}$ . Since the final component is the inner product  $S_I$ , we immediately get  $\hat{\mathbf{K}} = (\mathbf{y}_i^T\mathbf{y}_j)_{n\times n} = (\mathbf{\Lambda}_k^{1/2}\mathbf{V}^T)^T(\mathbf{\Lambda}_k^{1/2}\mathbf{V}^T) = \mathbf{V}\mathbf{\Lambda}_k\mathbf{V}^T = \mathbf{K}$  and  $\hat{\kappa}(\mathbf{y}_i, \mathbf{y}) = \mathbf{y}_i^T\mathbf{y} = \left((\mathbf{\Lambda}_k^{1/2}\mathbf{V}^T)^T(\mathbf{\Lambda}_k^{-1/2}\mathbf{V}^T\mathbf{t})\right)_i = (\mathbf{V}\mathbf{\Lambda}_k\mathbf{V}^T\mathbf{t})_i = \kappa(\mathbf{y}_i, \mathbf{y})$ .

## 4 Experimental Results

We evaluated our spectral kernels on two text data sets, namely Medline1033 and Reuters-21578, to demonstrate its applicability and effectiveness. We chose these two data sets for easy comparison with the experiments reported in [12]. Compared to the baseline, i.e., SVM classifier with linear kernel and without feature selection, our results were either better or on-par according to the  $F_1$  measure. The interested reader may refer to [20] for the complete details.

On the Medline1003, we configured a spectral kernel of the form  $SK\langle\Gamma_N, 1(a), S_I\rangle$ . We started with a small  $k$  feature space for the classifier with spectral kernel and increased the dimensionality until the classification performance deteriorated, i.e., when  $k > 250$ . The results of both `query23` and `query20` proved to be very encouraging. With a small  $k$  (less than 200), the spectral kernel  $SK\langle\Gamma_N, 1(a), S_I\rangle$  increased quickly to a result that was much better than baseline method according to  $F_1$  measure. In particular, for `query23`, the best performance delivered by the spectral kernel was 84.62%, almost twice that of the baseline method which was 42.11%.

On the Reuters21578, we configured the spectral kernel to  $SK\langle\Gamma_N, 1(b)2, S_I\rangle$ . While performing comparably on categories `earn` and `interest`, the spectral kernel

outperformed the linear kernel (i.e., baseline method) on the remaining eight categories with small  $k$  values (generally less than 300). In particular, on the `ship` category, the best  $F_1$  achieved by spectral kernel was 89.16% which is much higher than 82.05% delivered by linear kernel. More importantly, the  $F_1$  performance under most values of  $k$  were much higher than the baseline. This is an encouraging result showing the effectiveness of spectral kernels in text classification tasks.

Furthermore, eight of the performance plots on Reuters data set, and two of the performance plots on Medline data set showed that a small value of  $k$  (usually  $100 \leq k \leq 300$ ) is often sufficient to achieve good  $F_1$  performance. This observation is different from the selection of  $k$  in latent semantic kernels, where a larger  $k$  implied better performance (i.e., in the range of 500 to 1000 as observed in [12]). The small values of  $k$  is encouraging because they lead to shorter runtime to achieve the same classification accuracy as the latent semantic kernels.

The results on the Reuters data set did however reveal a shortcoming of the spectral kernel: there is no fix value of  $k$  that ensures consistent performance for different categories. This will be a practical limit that requires automated mechanisms to determine  $k$ . While we are working on this as part of our future work, using a single value of  $k$  enables us to compare the spectral kernel against the linear kernel objectively. As reported in [20], we achieved better performance than the baseline method when  $k = 270$ , and comparable performance when  $k = 120$ .

## 5 Conclusion and Future Work

We proposed a new kernel that uses the semantics extracted from spectral clustering. Unlike LSKs, spectral kernels are unique by the virtue of using spectral clustering information and its ability to support incremental updates to the kernel matrix that keeps the cost of training to a minimal. Further, we have shown that we can obtain the spectral embedding of both training and testing sets by matrix approximation. Hence, it is possible for spectral kernels to handle feature space of any dimensionality.

The closest piece of related work, to our knowledge, is the classification of projected  $k$ -dimensional space obtained by spectral clustering algorithms [21]. However, the proposal suffers from three drawbacks. First, if a new input arrives, there is a need to eigendecompose the new  $\mathbf{S}$  (which includes the new input) to obtain the new spectral space. This is time-consuming for classification during operation where large number of data may arrive. Second, since the spectral space is dependent on the testing set rather than the training set, the spectral space is unstable if the testing set is highly random. Third, the similarity relationship between training data is not fully exploited to significantly improve the accuracy of classification. Our work overcomes these drawbacks and provided a kernel framework of applying spectral clustering to classification.

As an attempt to develop a novel kernel for classification, we foresee much future work for research. In particular, our immediate interest is to be able to find a suitable value of  $k$  for each category in classification by means of automated mechanisms. This is important for practical reasons as maintaining the right value of  $k$  over the lifetime of classification can significantly improve classification accuracy.

## References

1. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
2. Scholkopf, B., Smola, A., Muller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* **10** (1998) 1299–1319
3. Bach, F.R., Jordan, M.I.: Kernel independent component analysis. *The Journal of Machine Learning Research* **3** (2003) 1–48
4. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
5. Li, W., Ng, W.K., Ong, K.L., Lim, E.P.: A spectroscopy of texts for effective clustering. In: Proc. 8th PKDD, Pisa, Italy (2004)
6. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. *J. American Society of Information Science* **41** (1990) 391–407
7. Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM* **46** (1999) 604–632
8. Lawrence, P., Sergey, B., Rajeev, M., Terry, W.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1999)
9. Siolas, G., d'Alché Buc, F.: Support vector machines based on a semantic kernel for text categorization. In: Proc. Int. Joint Conf. on Neural Networks (IJCNN). (2000) 5205–5210
10. Moschitti, A., Bejan, C.A.: A semantic kernel for predicate argument classification. In: Proc. Natural Language Learning (CoNLL), Boston, MA, USA (2004) 17–24
11. Gosselin, P.H., Cord, M.: Semantic kernel updating for content-based image retrieval. In: Proc. of IEEE 6th International Symposium on Multimedia Software Engineering (ISMSE), Miami, Florida (2004) 537–544
12. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent semantic kernels. In: In Proc. of 18th International Conference on Machine Learning (ICML), Williams College, US (2001) 66–73
13. Chung, F.R.K.: Spectral Graph Theory. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society (1997)
14. Ding, C.: Tutorial: Spectral clustering. In: Proc. of 21st International Conference on Machine Learning (ICML), Alberta, Canada (2004)
15. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (2000) 888–905
16. Maila, M., Shi, J.: A random walks view of spectral segmentation. In: Proc. of the 8th International Workshop on Artificial Intelligence and Statistics, Florida (2001)
17. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Proc. of Advances in Neural Information Processing Systems 14. (2001)
18. Papadimitriou, C.H., Tamaki, H., Raghavan, P., Vempala, S.: Latent semantic indexing: A probabilistic analysis. In: Proc. ACM PODS, Seattle, Washington, USA (1998) 159–168
19. Golub, G., Reinsch, C.: Handbook for Matrix Computation II, Linear Algebra. Springer-Verlag, New York (1971)
20. Li, W., Ong, K.L., Sun, A., Ng, W.K.: Spectral kernels for classification. Technical Report TRC05/05 (<http://www.deakin.edu.au/~leong/tr>), Deakin University (2005)
21. Kamvar, S.D., Klein, D., Manning, C.D.: Spectral learning. In: Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI). (2003)

# Data Warehousing and Knowledge Discovery: A Chronological View of Research Challenges

Tho Manh Nguyen<sup>1</sup>, A Min Tjoa<sup>1</sup>, and Juan Trujillo<sup>2</sup>

<sup>1</sup> Institute of Software Technology and Interactive Systems,  
Vienna University of Technology,  
Favoritenstr. 9-11/188, A-1040 Vienna, Austria  
{tho, amin}@ifs.tuwien.ac.at

<sup>2</sup> Dept. of Language and Information Systems, University of Alicante,  
Apto. Correos 99. E-03080. Spain  
jtrujillo@dlsi.ua.es

Data Warehousing and Knowledge Discovery has been widely accepted as a key technology for enterprises to improve their abilities in data analysis, decision support, and the automatic extraction of knowledge from data. Historically, the phrase knowledge discovery in databases was coined at the first KDD (Knowledge Discovery and Data Mining) workshop in 1989 to emphasize that knowledge is the end-product of a data-driven discovery process. Since then, much research has been accomplished in this field. This paper which is written as an epilogue of the DaWaK 2005 proceedings by the programme committee chairpersons together with Nguyen Manh Tho, should reflect the past development of DaWaK-results and other significant research outcomes in the area and above all should deliver a rough sketch of the current development and possible future work.

In the early 1990s, most businesses realized that there was an urgent need for more sophisticated tools for analyzing their business data, customer profiles and product information. Data mining and data warehousing technology mainly originated from these needs.

Data mining can be defined as the automated extraction of hidden predictive information from large amount of data. Data Mining is considered as an important step in the Knowledge Discovery -process that produces a particular enumeration of patterns (or models) over the data [9]. The most commonly used techniques in data mining and knowledge discovery in the late 1980s and early 1990s are artificial neural networks, decision trees, genetic algorithms, nearest neighbourhood, and rule induction [12,13].

William H. Inmon, who is widely accepted as the mental-father of data warehousing has been working on data warehousing concepts since 1983, and used for the first time this term in 1992 [10]. In 1993, Ted Codd coins the term OLAP (On-Line Analytical Processing) and defined the famous 12 OLAP rules [11]. Based on their definitions, a data warehouse is actually a comprehensive system that includes all the processes, tools and technologies necessary for extracting data from many operational, legacy, possibly heterogeneous data sources and managing them in a separate storage (warehouse) to provide end-user decision-support access. OLAP tools are well-suited for complex data analysis, such as multidimensional data analysis, and to assist in



decision support activities. The multidimensional (MD) data model has been proved to be the most suitable for OLAP applications.

The two sides of the coin for decision making are principally formed by Data Warehousing and Knowledge Discovery. Knowledge discovery in data warehouses focuses upon the extraction of interesting and previously unknown knowledge [3]. Researchers and application developers have designed knowledge discovery systems for a large number of application domains including finance, health, telecommunications and marketing. Consequently, many areas of research in data warehousing and knowledge discovery mushroomed in the late 1990s. Initially, when the concepts of OLAP and multi-dimensional databases have not yet the desired level of maturity, most researchers focus on the topics of data warehouse design, view selection and maintenance, multiple query optimization using views, on-line view maintenance, OLAP operators and environment, fragmentation of multidimensional database [1, 23, 24]. In the data mining community, interests are focused on data & web mining, pattern recognition and time series databases [1, 14, 15, 16]. More attention is also devoted to scientific data exploration dealing with such research objects as mining and discovery on biological data, spatial, text and multimedia data. Distributed and parallel mining techniques become more and more in use and some large-scale parallel and distributed knowledge discovery systems start to appear [17]. With the widespread of web applications web usage analysis and user profiling builds a special focus of investigations which huge relevance for e-commerce [17].

In the early 2000 the research community gradually solve the most basic issues in data warehouse design, materialized view maintenance and selection, OLAP query design and evolution. However, with the development of the increasing number of commercial products in OLAP and data warehousing, the data warehouse research activities force to concentrate towards more advanced techniques such as integrating active rules, update filtering, parallel processing, summarizability problem, data expiry, data indexing [2,3]. These activities include also the increasing consideration of security issues in OLAP and generally in the data warehousing environment [2, 25] - as well as the advanced interest on the heterogeneous and distributed data warehousing environment. [3,26]. The concept of object-orientation and the emerging XML technology cause significant implications on the design and development of data warehouse applications. [3, 26].

In the data mining and knowledge discovery research fields, the early 2000's witness new mining algorithms and techniques which are proposed and applied in a variety of applications such as text mining, outlier detection in scientific data, mining of temporal patterns, optimizing inventory in E-commerce, telephony and ISP applications [18]. Special attention is devoted to Web mining, interactive knowledge exploration, matchmaking and visualization [2, 18]. Mining of Web-log data, and multimedia data are still most important research topics while mining in bioinformatics has become an emerging field. [3,19].

In 2002 and 2003, with the advances of modern monitoring technologies (i.e. sensors, RFID, transmission of huge amount of digital satellite monitoring data) and with the demand of high speed business changes, the integration of a special type of data source namely of continuous data streams is becoming more and more essential. Data streams occur in applications such as sensor networks, networking flow analysis, web clicks stream analysis, telecommunication fraud detection, e-business and stock mar-

ket online analysis. One of the main characteristics of data streams is the impracticality of complete storage – hence we are usually restricted to the storage of samples or aggregations. It is demanding to conduct advanced analysis over fast and huge data streams to capture the trends, patterns, and exceptions. As a further consequence, the concept of near-real time data warehouses was initially announced in [4] while concurrently we still observe further intense investigations on parallel and distributed warehousing. [4, 27]. Ontology Structures, which are a foundation of the Semantic Web [7], has been applied among others for the integration of heterogeneous Data Quality Improving techniques [5]. With the increasing embedding of Web data as one of the main data sources, in DWH-research the Web-Warehousing concept is thus investigated in-depth together with its accompanying concepts and related technologies such as XML OLAP Cube, XML Warehouse, Warehouse design based on XML Schema [5, 28]. OLAP researchers proceed in the intense investigation of advanced improvements in Cube presentation, management, and performance (which obviously is not restricted on traditional data, but also includes XML data and other non-standard data sources such as spatial data [5, 28]).

Evidently the data mining community recognizes the important role of streams and time series analysis applications. A plethora of algorithms and techniques were proposed to mine high-speed data streams, to analyse click streams, and to correlate synchronised and asynchronised online streams [20, 21]. Furthermore, with the huge amount of web data, research conducted on web search tools and web classification applications builds a main focus [4, 20, 21]. Data mining in Bioinformatics, multimedia and complex data still receives a lot of interesting and research efforts [20, 21]. Data mining techniques are also applied to improve database-engine issues by using techniques which have been successfully deployed in other areas of applied computer science and systems theory – a prominent example herefore is the use of Rough Set Theory, as an alternative of fuzzy sets [5].

The complexity of existing data warehousing and enterprise systems has reached a new quality in the recent years. However, there is still a lack of comprehensive documentation and dissemination of requirement engineering methods. Therefore, conceptual modelling still plays an essential role in integrating higher level of abstractions for the description of processes all components of the data warehouse architecture [6]. Spatial data warehouses reach some maturity with the design framework of Geographical Dimensional Schema. OLAP-techniques are now enhanced with innovations in Range Aggregation and approximate queries answering [6, 29]. Data streams analysis and time series mining techniques receive an enduring boosted interest from the researchers [6, 22]. Pattern discovery and event sequence mining has emerged as a new field of interest while data semantics became an increasingly important issue. The topic of data visualization and exploration gets increased interests while traditional mining techniques such as association rules, clustering remain steadily prevailing [6, 22]. In parentheses it should be remarked that not only academic researchers, but increasingly the industrial community is concentrating in these activities [6, 22].

Due to the fact of the tremendous amount of existing data warehouse systems, more current efforts are taken to integrate and transform the different heterogeneous data warehousing systems. An important innovation can be observed in extending existing relevant and successful CASE tools used in software development (most notably UML-tools) for data warehouse design and development. With the expanding

spread of open source tools we can observe a trend to solve the Business Intelligence issues in the open source community.

Artificial intelligence techniques such as machine learning, neural network, case-based reasoning have inspired a continuous fast growing attention within the data mining community. An expanding interest can be observed in the integration of text processing and the mining unstructured data using data semantics approaches in combination with traditional techniques i.e. clustering, association rules, pattern recognition...

Although research in data warehousing and knowledge discovery has generated successful and remarkable results, new applications still continuously generate new challenges to the research community. With the exponential growing amount of information to be included in the decision making process, the data to be considered becomes more and more complex in both structure and semantics. Consequently, the process of retrieval and knowledge discovery from this huge amount of heterogeneous complex data builds the litmus-test for the research in the area. Current emerging real world applications such as real-time data warehousing, analysis of spatial and spatio-temporal data, OLAP mining, mobile OLAP and more recent applications in natural sciences (especially bioinformatics) requires novel representation and manipulation techniques for non-standard data and tailored efficient algorithms for the computation of dedicated aggregate queries and application-specific index structures.

Vendors like Oracle, IBM and Microsoft already tightly integrate OLAP and data mining in their DBMS commercial tools. Therefore, we are observing a trend to close the gap between data warehousing and data mining. It is even imaginable that in the medium-term future the separation of OLAP-data warehouses (with its semantic redundant storage requirement) from its OLTP-database-sources could be bridge by innovative view-generation techniques as originally proposed in the three-level architecture.

## References

- [1] Mukesh K. Mohania, A. Min Tjoa (Eds.): *Data Warehousing and Knowledge Discovery, First International Conference, DaWaK '99*, Florence, Italy, August 30 - September 1, 1999, Proceedings. LNCS 1676 Springer 1999, ISBN 3-540-66458-0
- [2] Yahiko Kambayashi, Mukesh K. Mohania, A. Min Tjoa (Eds.): *Data Warehousing and Knowledge Discovery, Second International Conference, DaWaK 2000*, London, UK, September 4-6, 2000, Proceedings. LNCS 1874 Springer 2000, ISBN 3-540-67980-4
- [3] Yahiko Kambayashi, Werner Winiwarter, Masatoshi Arikawa (Eds.): *Data Warehousing and Knowledge Discovery, Third International Conference, DaWaK 2001*, Munich, Germany, September 5-7, 2001, Proceedings. LNCS 2114 Springer 2001, ISBN 3-540-42553-5
- [4] Yahiko Kambayashi, Werner Winiwarter, Masatoshi Arikawa (Eds.): *Data Warehousing and Knowledge Discovery, 4th International Conference, DaWaK 2002*, Aix-en-Provence, France, September 4-6, 2002, Proceedings. LNCS 2454 Springer 2002, ISBN 3-540-44123-9
- [5] Yahiko Kambayashi, Mukesh K. Mohania, Wolfram Wöß (Eds.): *Data Warehousing and Knowledge Discovery, 5th International Conference, DaWaK 2003*, Prague, Czech Republic, September 3-5, 2003, Proceedings. LNCS 2737 Springer 2003, ISBN 3-540-40807-X

- [6] Yahiko Kambayashi, Mukesh K. Mohania, Wolfram Wöß (Eds.): *Data Warehousing and Knowledge Discovery, 6th International Conference, DaWaK 2004*, Zaragoza, Spain, September 1-3, 2004, Proceedings. LNCS 3181 Springer 2004, ISBN 3-540-22937-X
- [7] Tim Berners-Lee: *Semantic Web Road map*, September 1998.  
<http://www.w3.org/DesignIssues/Semantic.html>
- [8] Mukesh K. Mohania, Yahiko Kambayashi, A. Min Tjoa, Roland Wagner, Ladjel Bella-treche: *Trends in Database Research, Database and Expert Systems Applications, 12th International Conference, DEXA 2001* Munich, Germany, September 3-5, 2001
- [9] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth: *From Data Mining to Knowledge Discovery in Databases, American Association for Artificial Intelligence*, 0738-4602-1996
- [10] W. H. Inmon: *Building the Data Warehouse 1st edition*, 1992.
- [11] E.F. Codd & Associates : *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*, white paper ,1993
- [12] Usama M. Fayyad, Ramasamy Uthurusamy (Eds.): *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, Seattle, Washington, July 1994. AAAI Press, Technical Report WS-94-03, ISBN 0-929280-73-3
- [13] Usama M. Fayyad, Ramasamy Uthurusamy (Eds.): *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, Montreal, Canada, August 20-21, 1995. AAAI Press, ISBN 0-929280-82-2
- [14] Evangelos Simoudis, Jiawei Han, Usama M. Fayyad (Eds.): *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 1996, ISBN 1-57735-004-9
- [15] David Heckerman, Heikki Mannila, Daryl Pregibon (Eds.): *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, Newport Beach, California, USA, August 14-17, 1997. AAAI Press, 1997, ISBN 1-57735-027-8
- [16] Rakesh Agrawal, Paul E. Stolorz, Gregory Piatetsky-Shapiro (Eds.): *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, August 27-31, 1998, New York City, New York, USA. AAAI Press, 1998, ISBN 1-57735-070-7
- [17] Usama Fayyad, Surajit Chaudhuri, David Madigan (Eds.): *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 15-18, 1999, San Diego, CA, USA. ACM, 1999
- [18] Raghu Ramakrishnan, Sal Stolfo, Roberto Bayardo, Ismail Parsa (Eds.): *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, August 20-23, 2000, Boston, MA, USA. ACM, 2000
- [19] Doheon Lee, Mario Schkolnick, Foster Provost, Ramakrishnan Srikant (Eds.): *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, August 26-29, 2001, San Francisco, CA, USA. ACM, 2001
- [20] Osmar R. Zaïane, Randy Goebel, David Hand, Daniel Keim, Raymond Ng (Eds.): *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 23-26, 2002, Edmonton, Alberta, Canada. ACM 2002, ISBN 1-58113-567-X
- [21] Lise Getoor, Ted E. Senator, Pedro Domingos, Christos Faloutsos (Eds.): *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, August 24 - 27, 2003. ACM 2003, ISBN 1-58113-737-0
- [22] Won Kim, Ron Kohavi, Johannes Gehrke, William DuMouchel (Eds.): *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, Washington, USA, August 22-25, 2004. ACM 2004, ISBN 1-58113-888-1

- [23] Il-Yeol Song, Toby J. Teorey (Eds.) : *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP, DOLAP 1998*, Washington, D.C., United States November 02 - 07, 1998
- [24] Il-Yeol Song, Toby J. Teorey (Eds.) : *Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP, DOLAP 1999*, Kansas City, Missouri, United States November 02 - 06, 1999
- [25] Rokia Missaoui, Il-Yeol Song (Eds.): *Proceedings of the 3rd ACM international workshop on Data warehousing and OLAP, DOLAP 2000*, McLean, Virginia, United States November 06 - 11, 2000
- [26] Joachim Hammer (Eds.): *Proceedings of the 4th ACM international workshop on Data warehousing and OLAP, 2001*, Atlanta, Georgia, USA November 09 - 09, 2001
- [27] Dimitri Theodoratos, Il-Yeol Song (Eds.): *Proceedings of the 5th ACM international workshop on Data warehousing and OLAP, DOLAP 2002*, November 8, 2002, McLean, VA, Proceedings. ACM 2002, ISBN 1-58113-5904
- [28] Stefano Rizzi, Il-Yeol Song (Eds.): *Proceedings of the 6th ACM international workshop on Data warehousing and OLAP, DOLAP 2003*, New Orleans, Louisiana, USA November 07 - 07, 2003
- [29] Il-Yeol Song, Karen C. Davis (Eds.): *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, DOLAP 2004*, Washington, DC, USA, November 12-13, 2004, Proceedings. ACM 2004, ISBN 1-58113-977-2

# Author Index

- Abbadi, Amr El 179  
Adam, Nabil 157  
Agrawal, Divyakant 179  
Aguilar-Saborit, Josep 200  
Agyemang, Malik 285  
Alhajj, Reda 285  
Allen, Robert 326  
An, Aijun 254  
Angiulli, Fabrizio 509  
Aouiche, Kamel 64  
Atluri, Vijayalakshmi 157  
Azuma, Naoki 233
- Barker, Ken 285  
Bellatreche, Ladjel 115  
Bentayeb, Fadila 64, 85  
Berenguer, Gema 95  
Bernardino, Jorge 356  
Bertino, Elisa 418  
Boukhalfa, Kamel 115  
Boussaïd, Omar 64, 85
- Carreira, Paulo 136  
Casali, Alain 428  
Chen, Ding-Yi 368  
Chen, Enhong 458  
Chen, Xia 368  
Cicchetti, Rosine 428
- Darmont, Jérôme 64, 85  
Dellis, Evangelos 243  
Dong, Zhao Yang 368  
Ducrou, Jon 398
- Eder, Johann 1  
Eklund, Peter 398
- Fan, Hao 126  
Fankhauser, Peter 210  
Feng, Jianlin 378  
Fovino, Igor Nai 418  
Furfaro, Filippo 478
- Galhardas, Helena 136  
Gebski, Matthew 388, 498
- Gidófalvi, Győző 275  
Gorawski, Marcin 190
- Hamrouni, Tarek 346  
Han, Hyoil 32  
Hasan, K.M. Azharul 233  
Higuchi, Ken 233  
Hu, Xiaohua 326  
Hua, Ming 408  
Huang, Houkuan 168  
Huang, Xiangji 254
- Im, Kwang Hyuk 265
- Jeng, Rong 336
- Kerdprasop, Kittisak 488  
Kerdprasop, Nittaya 488  
Kim, Jinho 146  
Kim, Tae Hyun 265  
Koncilia, Christian 1  
Korherr, Birgit 53  
Kuroda, Masayuki 233
- Lakhal, Lotfi 428  
Larriba-Pey, Josep-L. 200  
Lee, Byung-Suk 146  
Lee, Wookey 146  
Lehti, Patrick 210  
Li, Hongsong 168  
Li, Hua-Gang 179  
Li, Tongshu 458  
Li, Wenyuan 520  
Li, Xue 368, 468  
Lin, Wen-Yang 336  
List, Beate 53  
Liu, Shijin 168  
Long, Hao 378  
Lopes, Antónia 136
- Madeira, Henrique 356  
Malczok, Rafal 190  
Mazzeo, Giuseppe M. 478  
Missaoui, Rokia 221  
Moon, Yang-Sae 146

- Morzy, Mikołaj 295, 448  
 Morzy, Tadeusz 295  
 Muntés-Mulero, Victor 200
- Naouali, Sami 221  
 Natwichai, Juggapong 468  
 Nemeč, Jaromir 22  
 Ng, Wee-Keong 520  
 Nguyen, Tho Manh 22, 530
- Ok, Soo-Ho 146  
 Ong, Kok-Leong 520  
 Orłowska, Maria 468
- Panella, Ivan 105  
 Park, Byung-Kwon 32  
 Park, Sang Chan 265  
 Pedersen, Torben Bach 74, 275  
 Pereira, João 136  
 Piattini, Mario 95  
 Pizzuti, Clara 509
- Qian, Tieyun 378
- Roddick, John F. 315  
 Romero, Rafael 95
- Sattayatham, Pairote 488  
 Sattler, Kai-Uwe 438  
 Seeger, Bernhard 243  
 Serrano, Manuel 95  
 Shang, Xuequn 438  
 Sheu, Phillip C-y 458  
 Shi, Baile 408  
 Simitsis, Alkis 43  
 Sirangelo, Cristina 478  
 Skiadopoulos, Spiros 43  
 Slimani, Yahya 346  
 Song, Il-Yeol 32, 326
- Song, Min 326  
 Stefanov, Veronika 53  
 Sun, Aixín 520
- Terrovitis, Manolis 43  
 Thomsen, Christian 74  
 Tjoa, A Min 530  
 Torlone, Riccardo 105  
 Trujillo, Juan 11, 95, 530  
 Tseng, Ming-Cheng 336  
 Tsuji, Tatsuo 233
- Vassiliadis, Panos 43  
 Vieira, Jorge 356  
 Vlachou, Akrivi 243
- Wang, Wei 408  
 Wang, Yuanzhen 378  
 Wiggisser, Karl 1  
 Winarko, Edi 315  
 Windisch, Martin 22  
 Wojciechowski, Marek 295, 448  
 Wong, Raymond K. 388, 498  
 Wormuth, Bastian 398  
 Wu, Xindong 305
- Yahia, Sadok Ben 346  
 Yao, Qingsong 254  
 Yu, Hailing 179  
 Yu, Songmei 157
- Zakrzewicz, Maciej 295, 448  
 Zhang, Chengqi 305  
 Zhang, Jilian 305  
 Zhang, Shichao 305  
 Zhang, Shouzhi 408  
 Zhou, Haofeng 408  
 Zubcoff, José Jacobo 11  
 Zuzarte, Calisto 200