

## International Editorial Review Board

Sourav S. Bhowmick, Nanyang Technological University, Singapore  
Gillian Dobbie, University of Auckland, New Zealand  
Vladimir Estivill-Castro, Griffith University, Australia  
Christie Ezeife, University of Windsor, Canada  
Alex A. Freitas, University of Kent, UK  
Xiaohua Hu, Drexel University, USA  
Stephan Kudyba, New Jersey Institute of Technology, USA  
Clement H.C. Leung, Victoria University, Australia  
Sanjay Kumar Madria, University of Missouri – Rolla Rolla, USA  
Yannis Manolopoulos, Aristotle University, Thessaloniki, Greece  
Torben Bach Pedersen, Aalborg University, Denmark  
Jian Pei, Simon Fraser University, Canada  
Gerd Stumme, University of Kassel, Germany  
Chih Jeng Kenneth Tan, OptimaNumerics Ltd., UK  
Yannis Theodoridis, University of Piraeus, Greece  
Marek Wojciechowski, Poznan University of Technology, Poland  
Yongqiao Xiao, Georgia College and State University, USA  
Mohammed J. Zaki, Rensselaer Polytechnic Institute, USA  
Shichao Zhang, University of Technology, Sydney, Australia  
Zhi-Hua Zhou, Nanjing University, China

### Special Issue Proposal

**IJDWM** publishes special issues on the recent trends in Data Warehousing and Mining and related issues. A short proposal containing the title of proposed special issue, tentative list of invited authors, suggested date of publication and a call-for-papers should be sent to [david.taniar@infotech.monash.edu.au](mailto:david.taniar@infotech.monash.edu.au) for review.

IJDWM is listed in the following indexes: Burrelle's Media Directory; Cabell's Directory; CSA Illumina; DEST Register of Refereed Journals; GetCited; The Index of Information Systems Journals; INSPEC; MediaFinder; Standard Periodical Directory; Ulrich's International Periodicals Directory



**IGI PUBLISHING**

Hershey • New York

Order online at [www.igi-global.com](http://www.igi-global.com) or call 717-533-8845 x100  
Mon-Fri 8:30 am - 5:00 pm (est) or fax 24 hours a day 717-533-8661

# INTERNATIONAL JOURNAL OF DATA WAREHOUSING AND MINING

January-March 2009, Vol. 5, No. 1

## Table of Contents

### SURVEY ARTICLE

- 1 A Survey on Temporal Data Warehousing**  
*Matteo Golfarelli, DEIS - University of Bologna, Italy*  
*Stefano Rizzi, DEIS - University of Bologna, Italy*

### RESEARCH ARTICLES

- 18 Lossless Reduction of Datacubes using Partitions**  
*Alain Casali, Aix-Marseille Universités, France*  
*Sébastien Nedjar, Aix-Marseille Universités, France*  
*Rosine Cicchetti, Aix-Marseille Universités, France*  
*Lotfi Lakhal, Aix-Marseille Universités, France*  
*Noël Novelli, Aix-Marseille Universités, France*
- 36 A Parameterized Framework for Clustering Streams**  
*Vasudha Bhatnagar, University of Delhi, India*  
*Sharanjit Kaur, University of Delhi, India*  
*Laurent Mignet, I.B.M., Indian Research Lab, India*
- 57 A Hybrid Method for High-Utility Itemsets Mining in Large High-Dimensional Data**  
*Guangzhu Yu, Donghua University, China*  
*Shihuang Shao, Donghua University, China*  
*Bin Luo, Guangdong University of Technology, China*  
*Xianhui Zeng, Donghua University, China*

# International Journal of Data Warehousing and Mining

## Guidelines for Manuscript Submissions

**Mission:** *International Journal of Data Warehousing and Mining (IJDWM)* aims to publish and disseminate knowledge on an international basis in the areas of data warehousing and data mining. It is published multiple times a year, with the purpose of providing a forum for state-of-the-art developments and research, as well as current innovative activities in data warehousing and mining. In contrast to other journals, this journal focuses on the integration between the fields of data warehousing and data mining, with emphasis on the applicability to real world problems. The journal is targeted at both academic researchers and practicing IT professionals.

**Coverage:** The journal is devoted to the publications of high quality papers on theoretical developments and practical applications in data warehousing and data mining. Original research papers, state-of-the-art reviews, and technical notes are invited for publications. The journal accepts paper submission of any work relevant to data warehousing and data mining. Special attention will be given to papers focusing on mining of data from data warehouses; integration of databases, data warehousing, and data mining; and holistic approaches to mining and archiving data. A summary of the scope of data warehousing and mining includes: data models; data structures; design data warehousing process; online analytical process; tools and languages; data mart and practical issues; data mining methods. For a full listing of coverage topics, please visit [www.igi-global.com/ijdwm](http://www.igi-global.com/ijdwm).

**Originality:** Prospective authors should note that only original and previously unpublished manuscripts will be considered. Furthermore, simultaneous submissions are not acceptable. Submission of a manuscript is interpreted as a statement of certification that no part of the manuscript is copyrighted by any other publication nor is under review by any other formal publication. It is the primary responsibility of the author to obtain proper permission for the use of any copyrighted materials in the manuscript, prior to the submission of the manuscript.

**Style:** Submitted manuscripts must be written in the APA (American Psychological Association) editorial style. References should relate only to material cited within the manuscript and be listed in alphabetical order, including the author's name, complete title of the cited work, title of the source, volume, issue, year of publication, and pages cited. Please do not include any abbreviations. See the following examples:

Example 1: Single author periodical publication.

Smith, A. J. (2002). Information and organizations. *Management Ideology Review*, 16(2), 1-15.

Example 2: Multiple authors periodical publication.

Smith, A. J., & Brown, C. J. (1988). Organizations and information processing. *Management Source*, 10(4), 77-88.

Example 3: Books.

Smith, A. J. (2002). *Information booklet*. New York: J.J. Press.

State author's name and year of publication where you use the source in the text. See the following:

Example 1:

In most organizations, information resources are considered a major resource (Brown, 1988; Smith, 2002).

Example 2:

Brown (2002) states that the value of information is recognized by most organizations.

Direct quotations of another author's work should be followed by the author's name, date of publication, and the page(s) on which the quotation appears in the original text.

Example 1:

Brown (1989) states that "the value of information is realized by most organizations" (p. 45).

Example 2:

In most organizations, "information resources are considered to be a major organization asset" (Smith, 2002, pp. 35-36) and must be carefully monitored by the senior management.

For more information please consult the APA manual.

**Review Process:** To ensure the high quality of published material, **IJDWM** utilizes a group of experts to review submitted manuscripts. Upon receipt of the manuscript, two reviewers are selected from the Editorial Review Board of the journal. The selection is based upon the particular area of expertise of the reviewers, matched to the subject matter of the submission. An additional ad-hoc reviewer is also selected to review the manuscript. Therefore, each submission is accordingly blind reviewed by at least three reviewers. Revised manuscripts will be reviewed again by the original review panel with the addition of one new reviewer. Return of a manuscript to the author(s) for revision does not guarantee acceptance of the manuscript for publication. The final decision will be based upon the comments of the reviewers, upon their second review of the revised manuscript.

**Copyright:** Authors are asked to sign a warranty and copyright agreement upon acceptance of their manuscript, before the manuscript can be published. All copyrights, including translation of the published material into other languages are reserved by the publisher, IGI Global. Upon transfer of the copyright to the publisher, no part of the manuscript may be reproduced in any form without written permission of the publisher.

**Submission:** Authors are asked to submit their manuscripts for possible publication by e-mail as a file attachment in Microsoft Word or RTF (Rich Text Format) to [david.taniar@infotech.monash.edu.au](mailto:david.taniar@infotech.monash.edu.au). The main body of the e-mail message should contain the title of the paper and the names and addresses of all authors. Manuscripts must be in English. The author's name should not be included anywhere in the manuscript, except on the cover page. Manuscripts must also be accompanied by an abstract of 100-150 words, precisely summarizing the mission and object of the manuscript.

**Length:** The length of the submitted manuscript is not specifically limited, however, the length should be reasonable in light of the chosen topic. Discussion and analysis should be complete but not unnecessarily long or repetitive.

**Correspondence:** The acknowledgment letter regarding the receipt of the manuscript will be promptly sent. The review process will take approximately 8-16 weeks, and the author will be notified concerning the possibility of publication of the manuscript as soon as the review process is completed. All correspondence will be directed to the first author of multi-authored manuscripts. It is the responsibility of the first author to communicate with the other author(s). Authors of accepted manuscript will be asked to provide a final copy of their manuscript in either Word or RTF text format stored on a 3 1/2" disk, zip disk or CD-ROM, accompanied by a hardcopy of the manuscript and the original signed copy of the Warranty and Copyright Agreement. The accepted manuscript will be edited by the Journal copyeditor for format and style.

**Book Reviews:** **IJDWM** invites prospective book reviewers to submit their review of either textbooks or professional books for possible inclusion in the journal. Reviewers should focus on the following guidelines when developing the book review:

- Book reviews must not exceed 1,500 words.
- Reviews should summarize the book and indicate the highlights, strengths, and weaknesses of the book.
- Reviews should evaluate the organizational and managerial applications of the material discussed in the book relevant to information resources and technology management.
- Reviews should critique and constructively evaluate the author's work and not merely list the chapters' contents.
- The writing style, accuracy, relevance, and the need for such a work in the discipline should be analyzed.
- The review must include the title of the book, author, publishing company, publication date, number of pages, cost (if listed), and ISBN number.
- Each submission must be accompanied by a short biography of the reviewer.

**Case Studies:** The **IJDWM** encourages submissions of case studies based on actual scenarios related to practice-based issues and practical applications of data warehousing and mining. Case studies must not exceed 2000 words and must provide adequate information regarding the educational environment upon which the study is based, presentation of the issues involved, coverage of any experiments or techniques involved, and elaborations of the lessons learned or conclusions drawn from the study.

ALL SUBMISSIONS AND QUESTIONS SHOULD BE FORWARDED TO:

David Taniar, Editor-in-Chief  
School of Business Systems, Monash University  
Clayton, Victoria, 3800, Australia  
[david.taniar@infotech.monash.edu.au](mailto:david.taniar@infotech.monash.edu.au)

# CALL FOR ARTICLES

## *International Journal of Data Warehousing and Mining*

*An official publication of the Information Resources Management Association!*

Prospective authors are invited to submit manuscripts for consideration for publication in the *International Journal of Data Warehousing and Mining*. IJDWM publishes original material concerned with all aspects of data warehousing and mining. The journal invites both conceptual and empirical manuscripts of high quality not currently under review by another publication.

### MISSION:

The IJDWM aims to publish and disseminate knowledge on an international basis in the areas of data warehousing and data mining. It is published multiple times a year, with the purpose of providing a forum for state-of-the-art developments and research, as well as current innovative activities in data warehousing and mining. In contrast to other journals, this journal focuses on the integration between the fields of data warehousing and data mining, with emphasis on the applicability to real world problems. The journal is targeted at both academic researchers and practicing IT professionals.

### COVERAGE/MAJOR TOPICS:

- data models
- data structures
- data mart
- mining databases
- and more! See [www.igi-global.com/ijdwm](http://www.igi-global.com/ijdwm)



ISSN 1548-3924

eISSN 1548-3932

Published quarterly

Full submission guidelines  
available at:  
<http://www.igi-global.com>

**All submissions should be emailed to: [david.taniar@infotech.monash.edu.au](mailto:david.taniar@infotech.monash.edu.au)**

Now when your institution's library subscribes to any IGIP journal, it receives the print version as well as the electronic version for one inclusive price. For information contact a customer service representative at [cust@igi-global.com](mailto:cust@igi-global.com) or 717/533-8845, ext. 100

Receive a **FREE JOURNAL SUBSCRIPTION** when you join the Information Resource Management Association! Choose any of our journals to receive free with your paid membership to IRMA and receive additional discounts for further journal subscriptions. For more information please visit [www.irma-international.org](http://www.irma-international.org).

All institutional subscriptions include free online access!

Please contact [cust@igi-global.com](mailto:cust@igi-global.com) for more information.



**IGI PUBLISHING**

701 E. Chocolate Ave., Suite 200

Hershey, PA 17033-1240, USA

Tel: 717/533-8845

Fax 717/533-8661

**PLEASE RECOMMEND THIS JOURNAL TO YOUR LIBRARY!**

# New for 2009!

I  
G  
J  
o  
u  
r  
n  
a  
l  
s



The goal of the *International Journal of Agent Technologies and Systems* is to increase awareness and interest in agent research, encourage collaboration and give a representative overview of the current state of research in this area. It aims at bringing together not only scientists from different areas of computer science, but also researchers from different fields studying similar concepts. The journal will serve as an inclusive forum for discussion on ongoing or completed work in both theoretical and practical issues of intelligent agent technologies and multi-agent systems.

The *International Journal of Agent Technologies and Systems* focuses on all aspects of agents and multi-agent systems, with a particular emphasis on how to modify established learning techniques and/or create new learning paradigms to address the many challenges presented by complex real-world problems.



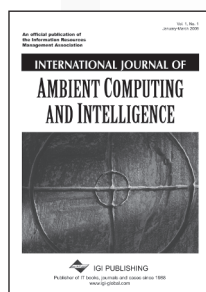
The objective of the *International Journal of E-Services and Mobile Applications* is to be a truly interdisciplinary journal providing comprehensive coverage and understanding of all aspects of e-services, self-services and mobile communication from different fields including marketing, management, and MIS. The journal invites contributions that are both empirical and conceptual, and is open to all types of research methodologies both from academia and industry.



The *International Journal of Sociotechnology and Knowledge Development* wishes to publish papers that offer a detailed analysis and discussion on sociotechnical philosophy and practices which underpin successful organizational change thus building a more promising future for today's societies and organizations.

It will encourage interdisciplinary texts that discuss current practices as well as demonstrating how the advances of - and changes within - technology affect the growth of society (and vice versa).

The aim of the journal is to bring together the expertise of people who have worked practically in a changing society across the world for people in the field of organizational development and technology studies including information systems development and implementation.



In an ambient intelligence world, devices work in concert to support people in carrying out everyday life activities and tasks in a natural way using information and intelligence that is hidden in the network connecting these devices. The *International Journal of Ambient Computing and Intelligence* will specifically focus on the convergence of several computing areas. The first is ubiquitous computing which focuses on self-testing and self repairing software, privacy ensuring technology and the development of various ad hoc networking capabilities that exploit numerous low-cost computing devices. The second key area is intelligent systems research, which provides learning algorithms and pattern matchers, speech recognition and language translators, and gesture classification and situation assessment. Another area is context awareness which attempts to track and position objects of all types and represent objects' interactions with their environments. Finally, an appreciation of human-centric computer interfaces, intelligent agents, multimodal interaction and the social interactions of objects in environments is essential.



**IGI PUBLISHING**

Order online at [www.igi-global.com](http://www.igi-global.com) or call 717.533.8845 ext.100  
Mon-Fri 8:30am-5:00pm (EST) or fax 24 hours a day 717.533.8661

## SURVEY ARTICLE

# A Survey on Temporal Data Warehousing

Matteo Golfarelli, DEIS - University of Bologna, Italy

Stefano Rizzi, DEIS - University of Bologna, Italy

---

## ABSTRACT

*Data warehouses are information repositories specialized in supporting decision making. Since the decisional process typically requires an analysis of historical trends, time and its management acquire a huge importance. In this paper we consider the variety of issues, often grouped under term temporal data warehousing, implied by the need for accurately describing how information changes over time in data warehousing systems. We recognize that, with reference to a three-levels architecture, these issues can be classified into some topics, namely: handling data/schema changes in the data warehouse, handling data/schema changes in the data mart, querying temporal data, and designing temporal data warehouses. After introducing the main concepts and terminology of temporal databases, we separately survey these topics. Finally, we discuss the open research issues also in connection with their implementation on commercial tools.*

**Keywords:** *business intelligence; data mart; data warehouse; multidimensional database design; spatiotemporal database*

---

## INTRODUCTION

At the core of most business intelligence applications, *data warehousing systems* are specialized in supporting decision making. They have been rapidly spreading within the industrial world over the last decade, due to their undeniable contribution to increasing the effectiveness and efficiency of the decisional processes within business and scientific domains. This wide diffusion was supported by remarkable research results aimed at improving querying performance, at refining the quality of data, and

at outlining the design process, as well as by the quick advancement of commercial tools.

In the remainder of the paper, for the sake of terminological consistency, we will refer to a classic architecture for data warehousing systems, illustrated in Figure 1, that relies on three levels:

1. The *data sources*, that store the data used for feeding the data warehousing systems. They are mainly corporate operational databases, hosted by either relational or legacy platforms, but in some cases they



may also include external web data, flat files, spreadsheet files, etc.

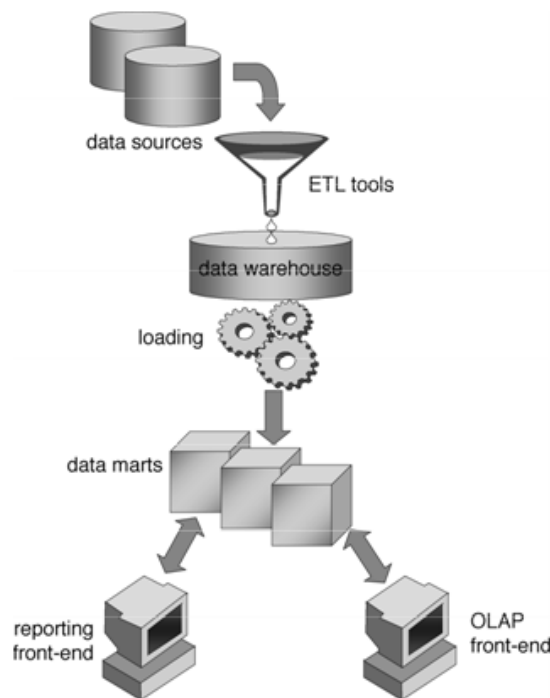
2. The *data warehouse* (also called *reconciled data level*, *operational data store* or *enterprise data warehouse*), a normalized operational database that stores detailed, integrated, clean and consistent data extracted from data sources and properly processed by means of ETL tools.
3. The *data marts*, where data taken from the data warehouse are summarized into relevant information for decision making, in the form of *multidimensional cubes*, to be typically queried by OLAP and reporting front-ends.

Cubes are structured according to the *multidimensional model*, whose key concepts are fact, measure and dimension. A *fact* is a focus of interest for the decisional process; its occurrences correspond to *events* that dynamically occur within the business world. Each event

is quantitatively described by a set of numerical *measures*. In the multidimensional model, events are arranged within an n-dimensional space whose axes, called *dimensions* of analysis, define different perspectives for their identification. Dimensions commonly are discrete, alphanumeric attributes that determine the minimum granularity for analyzing facts. Each dimension is the root of a (*roll-up*) *hierarchy* that includes a set of *levels*, each providing a way of selecting and aggregating events. Each level can be described by a set of *properties*.

As a consequence of the fact that the decisional process typically relies on computing historical trends and on comparing snapshots of the enterprise taken at different moments, one of the main characterizations of data warehousing systems is that of storing historical, non volatile data. Thus, time and its management acquire a huge importance. In this paper we discuss the variety of issues, often grouped under term *temporal data warehousing*, implied by the need for

Figure 1. Three-levels architecture for a data warehousing system





accurately describing *how information changes over time*. These issues, arising by the never ending evolution of the application domains, are even more pressing today, as several mature implementations of data warehousing systems are fully operational within medium to large business contexts. Note that, in comparison with operational databases, temporal issues are more critical in data warehousing systems since queries frequently span long periods of time; thus, it is very common that they are required to cross the boundaries of different versions of data and/or schema. Besides, the criticality of the problem is obviously higher for systems that have been established for a long time, since unhandled evolutions will determine a stronger gap between the reality and its representation within the database, which will soon become obsolete and useless (Golfarelli et al, 2006).

So, not surprisingly, there has been a lot of research so far regarding temporal issues in data warehousing systems. Basically, the approaches devised in the literature can be accommodated in the following (sometimes overlapping) categories:

- *Handling changes in the data warehouse* (discussed in the third section). This mainly has to do with maintaining the data warehouse in sync with the data sources when changes on either of these two levels occur.
- *Handling data changes in the data mart* (fourth section). Events are continuously added to data marts; while recorded events are typically not subject to further changes, in some cases they can be modified to accommodate errors or late notifications of up-to-date values for measures. Besides, the instances of dimensions and hierarchies are not entirely static.
- *Handling schema changes in the data mart* (fifth section). The data mart structure may change in response to the evolving business requirements. New levels and measures may become necessary, while others may become obsolete. Even the set of dimen-

sions characterizing a fact may be required to change.

- *Querying temporal data* (sixth section). Querying in presence of data and schema changes require specific attention, especially if the user is interested in formulating queries whose temporal range covers different versions of data and/or schema.
- *Designing temporal data warehouses* (seventh section). The specific characteristics of temporal data warehouses may require ad hoc approaches for their design, especially from the conceptual point of view.

The paper outline is completed by the second section, that introduces the main concepts and terminology of temporal databases, and by the eighth section, that summarizes some open issues and draws the conclusions.

## TEMPORAL DATABASES

Databases where time is not represented are often called *transient databases*. Within a transient database, only the current representation of real-world objects is stored and no track of changes is kept, so it is impossible to reconstruct how the object was in the past. Conversely, *temporal databases* focus on representing the inherent temporal nature of objects through the time-dependent recording of their structure and state. Two different time dimensions are normally considered in temporal databases, namely *valid time* and *transaction time* (Jensen et al., 1994). Valid time is the “real-world time”, i.e., it expresses the time when a fact is true in the business domain. Transaction time is the “database system time”, i.e., it expresses the time when facts are registered in the database. Temporal database systems are called *valid-time databases*, *transaction-time databases* or *bi-temporal databases* depending on their capacity to handle either or both of these two time dimensions (Tansel et al., 1993). The main benefit of using a bi-temporal database is that not only the history of the changes an object is subject to is recorded, but it is also

possible to obtain the same result from a query independently of the time when it is formulated (which might not happen if transaction time is not properly represented).

In the real world, objects change in both their state and their structure. This means that, within a database, both the values of data and their schema may change. Obviously, values of data are constantly modified by databases applications. On the other hand, modifying the database schema is a less frequent, though still common, occurrence in database administration. With reference to changes in the database schema, the literature commonly distinguishes three possibilities (Roddick, 1995):

- *Schema modification* is supported when a database system allows changes to the schema definition of a populated database, which may lead to loss of data.
- *Schema evolution* is supported when a database system enables the modification of the database schema without loss of existing data.
- *Schema versioning* is supported when a database system allows the accessing of all data, both retrospectively and prospectively, through user-definable version interfaces.

The significant difference between evolution and versioning is that the former does not require the maintenance of a schema history, while in the latter all past schema versions are retained. Note that, in the context of schema evolution and versioning, most authors agree that there is no need to distinguish valid time from transaction time (McKenzie & Snodgrass, 1990).

On the language side, TSQL2 (Snodgrass, 1995) is the most noticeable attempt to devise a query language for relational temporal databases. TSQL2 is a temporal extension to the SQL-92 language standard, augmented to enable users to specify valid-time and transaction-time expressions for data retrieval. As to querying in presence of schema versioning, while TSQL2 only allows users to punctually

specify the schema version according to which data are queried, other approaches also support queries spanning multiple schema versions (Grandi, 2002).

The concepts introduced in this section were originally devised for operational databases, and in particular for relational databases. While in principle they can also be applied to data warehousing systems, that in a ROLAP implementations are based on relational databases, the peculiarities of the multidimensional model and the strong relevance of time in the OLAP world call for more specific approaches.

## HANDLING CHANGES IN THE DATA WAREHOUSE

When considering temporal data, it is first of all necessary to understand how time is reflected in the database, and how a new piece of information affects existing data. From this point of view, Devlin (1997) proposes the following classification:

- *Transient data*: alterations and deletions of existing records physically destroy the previous data content.
- *Periodic data*: once a record is added to a database, it is never physically deleted, nor is its content ever modified. Rather, new records are added to reflect updates or deletions. Periodic data thus represent a complete record of the changes that have occurred in the data.
- *Semi-periodic data*: in some situations, due to performance and/or storage constraints, only the more recent history of data changes is kept.
- *Snapshot data*: a data snapshot is a stable view of data as it exists at some point in time, not containing any record of the changes that determined it. A series of snapshots can provide an overall view of the history of an organization.

Data sources normally adopt either a transient or a (semi-)periodic approach, depending

on whether the application domains requires keeping history of past data or not. The historical depth of a data warehouse is typically not less than the one of its data sources, thus data warehouses more often contain periodic data. Conversely, data marts normally conform to the snapshot model.

In order to model historical data in the data warehouse, Abello and Martín (2003) propose a bi-temporal storage structure where each attribute is associated to two couples of timestamps, so as to track the history of its values according to both valid and transaction time. Each attribute, or each set of attributes having the same behaviour with reference to changes (i.e., such that whenever an attribute in the set changes its value, all the others change too), is stored in a separate table so that a change occurred to one concept does not affect the other concepts. Obviously, such normalized and time-oriented structure is not suited for querying, that will take place on denormalized data marts fed from the data warehouse.

Since the data warehouse can be thought of as a set of derived, materialized views defined over a set of source schemata, the problem of evolving the content and the schema of derived views in connection to the source changes is highly relevant in the context of temporal data warehouses. Bellahsene (2002) distinguishes two subproblems: view maintenance and view adaptation.

*View maintenance* consists in maintaining a materialized view in response to data modifications of the source relations. Considering the width of the problem, we refer the reader to Gupta & Mumick (1995) for a taxonomy of view maintenance problems and a description of the main techniques proposed in the literature. A specific issue in view maintenance is how to provide temporal views over the history of source data, that may be non-temporal. We mention two approaches in this direction. Yang & Widom (1998) describe an architecture that uses incremental techniques to automatically maintain temporal views over non-temporal source relations, allowing users to ask temporal queries on these views. De Amo & Halfeld

Ferrari Alves (2000) present a self-maintainable temporal data warehouse that, besides a set of temporal views, includes a set of auxiliary relations containing only temporal information. Such auxiliary relations are used to maintain the data warehouse without consulting the source databases and to avoid storing the entire history of source databases in the warehouse.

*View adaptation* consists in recomputing a materialized view in response to changes either in the schema of the source relations or in the definition of the view itself. Changes in the source schemata may be due to an evolution of the application domain they represent, or to a new physical location for them. Changes in the definition of the view (i.e., in the data warehouse schema) may also be due to new requirements of the business users who query the data marts fed by the data warehouse. Among the approaches in this direction we mention the one by Bellahsene (1998), who proposes an extended relational view model to support view adaptation, aimed at maintaining data coherence and preserving the validity of the existing application programs. Performing a schema change leads to creating a new view, by means of an extended view definition language that incorporates two clauses: *hide*, which specifies a set of attributes to be hidden, and *add*, that allows a view to own additional attributes that do not belong to source relations. In the EVE framework (Lee, Nica, & Rundensteiner, 2002), in order to automate the redefinition of a view in response to schema changes in the data sources, the **database administrator** is allowed to embed her preferences about view evolution into the view definition itself. The preference-based view rewriting process, called *view synchronization*, identifies and extracts appropriate information from other data sources as replacements of the affected components of the original view definition, in order to produce an alternative view that somehow preserves the original one. Finally, the DyDa framework (Chen, Zhang, & Rundensteiner, 2006) supports compensating queries, that cope with erroneous results in view maintenance due to concurrent

updates in data source, in presence of data and schema changes.

The key idea of adaptation techniques is to avoid recomputing the materialized view from scratch by relying on the previous materialization and on the source relations. For instance, Bellahsene (2002) focuses on the adaptation of the data warehouse in response to schema changes arising on source relations located on multiple sites. To adapt the extent of the data warehouse in response to these changes, she adopts rewriting algorithms that make use of containment checking, so that only the part of the new view that is not contained in the old view will be recomputed. In the same context, a distinctive feature of the *AutoMed* system (Fan & Poulouvassilis, 2004) is the capability of handling not only schema evolutions in materialized data integration scenarios, but also changes in the data model in which the schema is expressed (e.g., XML vs. relational). This is achieved by applying sequences of primitive transformations to a low-level hypergraph-based data model, in whose terms higher-level modeling languages are defined.

With reference to the problem of keeping the data warehouse in sync with the sources, Wrembel and Bebel (2007) propose a metamodel for handling changes in the operational data sources, which supports the automatic detection of structural and content changes in the sources and their automatic propagation to the data warehouse.

Finally, Combi & Oliboni (2007) focus on the management of time-variant semi-structured XML data within the data warehouse. In particular, they propose a representation based on graphs whose nodes denote objects or values and are labeled with their validity interval; the constraints related to correct management of time are then discussed.

## HANDLING DATA CHANGES IN THE DATA MART

Content changes result from user activities that perform their day-to-day work on data sources

by means of different applications (Wrembel & Bebel, 2007). These changes are reflected in the data warehouse and then in the data marts fed from it.

The multidimensional model provides direct support for representing the sequence of events that constitute the history of a fact: by including a temporal dimension (say, with date granularity) in the fact, each event is associated to its date. For instance, if we consider an *ORDER* fact representing the quantities in the lines of orders received by a company selling PC consumables, the dimensions would probably be *product*, *orderNumber*, and *orderDate*. Thus, each event (i.e., each line of order) would be associated to the ordered product, to the number of the order it belongs to, and to the order date.

On the other hand, the multidimensional model implicitly assumes that the dimensions and the related levels are entirely static. This assumption is clearly unrealistic in most cases; for instance, considering again the order domain, a company may add new categories of products to its catalog while others can be dropped, or the category of a product may change in response to the marketing policy.

Another common assumption is that, once an event has been registered in a data mart, it is never modified so that the only possible writing operation consists in appending new events as they occur. While this is acceptable for a wide variety of domains, some applications call for a different behavior; for example the quantity of a product ordered in a given day could be wrongly registered or could be communicated after the ETL process has run.

These few examples emphasize the need for a correct handling of changes in the data mart content. Differently from the problem of handling schema changes, the issues related to data changes have been widely addressed by researchers and practitioners, even because in several cases they can be directly managed in commercial DBMSs. In the following subsections we separately discuss the issues related to changes in dimensional data and factual data, i.e., events.

## Changes in Dimensional Data

By this term we mean any content change that may occur within an instance of a hierarchy, involving either the dimension itself, or a level, or a property. For instance, considering a product hierarchy featuring levels *type* and *category*, the name of a product may change, or a new category may be introduced so that the existing types have to be reassigned to categories.

The study of changes in dimensional data has been pioneered by Kimball (1996), who coined the term *slowly-changing dimension* to point out that, differently from data in fact tables, changes within the dimension tables occur less frequently. He proposed three basic modeling solutions for a ROLAP implementation of the multidimensional model, each inducing a different capability of tracking the history of data. In the *Type I* solution he simply proposes to overwrite old tuples in dimension tables with new data: in this case, tracking history is not possible but changes in the hierarchy data keep the data mart up-to-date. Conversely, in the *Type II* solution, each change produces a new record in the dimension table: old events stay related to the old versions of hierarchies, while new events are related to the current version. In order to allow two or more tuples representing the same hierarchy instance to be included in the dimension table, surrogate keys must necessarily be adopted. Finally, the *Type III* solution is based on augmenting the schema of the dimension table by representing both the current and the previous value for each level or attribute subject to change.

Other solutions, based on these basic ones, have been proposed over time. In particular, a complete historicization of the dimension tables determines higher expressivity. This can be obtained for instance as an extension of Type II, by adding to the dimension table schema a couple of timestamps storing the validity interval for each tuple, plus an attribute storing the surrogate key of the first version of the tuple. This solution is sometimes called *Type VI* (I+II+III) since it covers all the previous ones.

The solutions discussed so far have different querying capabilities; with reference to the terminology proposed by SAP (2000), three main querying scenarios can be distinguished:

- *Today is yesterday*: all events are related to the current value of the hierarchy. This scenario is supported by all the discussed solutions.
- *Today or Yesterday*: each event is related to the hierarchy value that was valid when the event occurred. This scenario, that reconstructs the historical truth, is supported by Type II and VI solutions.
- *Yesterday is Today*: each event is related to the hierarchy value that was valid at a given time in the past. This scenario is supported by Type VI solution only.

Other solutions for handling changes in dimensional data have been devised thereafter. Two relevant proposals, that study the problem from a more conceptual point of view, are by Bliujute et al. (1998) and Pedersen and Jensen (1999). The first one proposes a temporal star schema that, differently from the traditional one, omits the time dimension table and timestamps each row in every table instead, treating the fact table and the dimension tables equally with respect to time. Similarly, the second one proposes to handle changes by adding timestamps to all the components of a multidimensional schema: the values of both dimensions and facts, the inter-level partial order that shapes hierarchy instances and the fact-dimension relationships. Another model that supports changes in data by timestamping dimensional data is COMET (Eder, Koncilia, & Morzy, 2002), that also supports schema versioning using a fully historicized meta-model. Finally, Chamoni and Stock (1999) suggest to couple the multidimensional cube with meta-cubes that store dimension structures together with their timestamps.

A model supporting data changes should be coupled with meaningful operators to carry them out. An interesting proposal in this



direction comes from Hurtado, Mendelzon, & Vaisman (1999b), who introduces a set of high-level operators based on sequences of elemental operators (Hurtado, Mendelzon, & Vaisman, 1999a) for both schema and data changes. The operators for data changes are *reclassify*, that changes the roll-up partial order between levels, *split*, that reorganizes a hierarchy after one instance has been replaced by two or more ones, *merge*, that merges two instances of a hierarchy into a single one, and *update*, that simply changes the value of an instance without affecting the roll-up partial order. Since changes to hierarchy instances could affect summarizability, the definition of models and operators is usually coupled with a set of constraints aimed at enforcing data consistency (Hurtado, Mendelzon, & Vaisman, 1999b; Eder, Koncilia, & Morzy, 2002; Letz, Henn, & Vossen, 2002).

## Changes in Factual Data

We start this section by preliminarily mentioning the two basic paradigms introduced by Kimball (1996) for representing inventory-like information in a data mart: the *transactional model*, where each increase and decrease in the inventory level is recorded as an event, and the *snapshot model*, where the current inventory level is periodically recorded. A similar characterization is proposed by Bliujute et al. (1998), who distinguish between *event-oriented data*, like sales, inventory transfers, and financial transactions, and *state-oriented data*, like unit prices, account balances, and inventory levels. This has been later generalized to define a classification of facts based on the conceptual role given to events (Golfarelli & Rizzi, 2007b):

- *Flow facts* (*flow measures* in Lenz & Shoshani, 1997) record a single transaction or summarize a set of transactions that occur during the same time interval; they are monitored by collecting their occurrences during a time interval and are cumulatively measured at the end of that period. Examples of flow facts are orders and enrollments.
- *Stock facts* (*stock measures* in Lenz & Shoshani, 1997) refer to an instant in time and are evaluated at that instant; they are monitored by periodically sampling and measuring their state. Examples are the price of a share and the level of a river.

By the term *changes in factual data* we mean any content change an event may be subject to, involving either the values of its measures or the dimensional elements it is connected to. Changes in factual data are a relevant issue in all those cases where the values measured for a given event may change over a period of time, to be consolidated only after the event has been for the first time registered in the data mart. These *late measurements* typically happens when the early measurements made for events are subject to errors (e.g., the amount of an order may be corrected after the order has been registered) or when events inherently evolve over time (e.g., notifications of university enrollments may be received and registered several days after they were issued). This problem becomes even more evident as the timeliness requirement takes more importance (Jarke, Jeusfeld, Quix, & Vassiliadis, 1999). This is the case for *zero-latency* data warehousing systems (Bruckner & Tjoa, 2002), whose goal is to allow organizations to deliver relevant information as fast as possible to knowledge workers or decision systems that need to react in near real-time to new information.

In these contexts, if the up-to-date state is to be made timely visible to the decision makers, past events must be continuously updated to reflect the incoming late measurements. Unfortunately, if updates are carried out by physically overwriting past registrations of events, some problems may arise. In fact, accountability and traceability require the capability of preserving the exact information the analyst based her decision upon. If the old registration for an event is replaced by its latest version, past decisions can no longer be justified. Besides, in some applications, accessing only up-to-date versions of information is not sufficient to ensure the correctness of analysis. A typical case is that of queries requiring to compare the progress of an

ongoing phenomenon with past occurrences of the same phenomenon: since the data recorded for the ongoing phenomenon are not consolidated yet, comparing them with past consolidated data may not be meaningful (Golfarelli & Rizzi, 2007b).

Supporting accountability and traceability in presence of late measurements requires the adoption of a bi-temporal solution where both valid and transaction time are represented by means of timestamps. Only few approaches in the literature are specifically focused on studying this specific topic. Kimball (2000) states that a bi-temporal solution may be useful to cope with late measurements. Bruckner & Tjoa (2002) discuss the problem of temporal consistency in consequence of delayed discovery of real-world changes and propose a solution based on valid time, revelation time and loading time. Loading time is the point in time when a new piece of information is loaded in the data mart, while revelation time is the point in time when that piece of information was realized by at least one data source. Finally, Golfarelli & Rizzi (2007b) propose to couple valid time and transaction time and distinguish two different solutions for managing late measurements: *delta solution*, where each new measurement for an event is represented as a delta with respect to the previous measurement, and transaction time is modeled by adding to the schema a new temporal dimension to represent when each registration was made in the data mart; and *consolidated solution*, where late measurements are represented by recording the consolidated value for the event, and transaction time is modeled by two temporal dimensions that delimit the time interval during which each registration is current.

## HANDLING SCHEMA CHANGES IN THE DATA MART

According to (Wrembel & Bebel, 2007), schema changes in the data mart may be caused by different factors:

- Subsequent design iterations in the context of an incremental approach to data mart design.
- Changes in the user requirements, triggered for instance by the need for producing more sophisticated reports, or by new categories of users that subscribe to the data mart.
- Changes in the application domain, i.e., arising from modifications in the business world, such as a change in the way a business is done, or a changing in the organizational structure of the company.
- New versions of software components being installed.
- System tuning activities.

For instance, it may be necessary to add a subcategory level to the product hierarchy to allow more detailed analysis, or to add a measure  $\text{revenueInEuro}$  due to the introduction of a new currency.

As stated in the second section, depending on how previous schema versions are managed, two main classes of approaches may be distinguished: *schema evolution*, that allows modifications of the schema without loss of data but does not maintain the schema history, and *schema versioning*, where past schema definitions are retained so that all data may be accessed through a version specified by the user. In the two following subsection these two classes of approaches will be separately surveyed.

### Evolution

The main problem here is to support a set of operators for changing the data mart schema, while enabling lossless migration of existing data from the past schema version to the new one.

In this context, FIESTA is a methodology where the evolution of multidimensional schemata is supported on a conceptual level, thus for both ROLAP and MOLAP implementations (Blaschka, Sapia, & Höfling, 1999; Blaschka, 2000). Core of the approach is a schema evolution algebra which includes a formal multidimensional data model together with a wide set



of schema evolution operations, whose effects on both schema and instances are described. Essentially, the operations allow dimensions, hierarchy levels, properties and measures to be added and deleted from the multidimensional schema. Since OLAP systems are often implemented on top of relational DBMSs, the approach also shows how a multidimensional schema can be mapped to a relational schema by means of a meta-schema that extends the catalogue of the underlying DBMS. Each sequence of evolution operations is then transformed into a sequence of relational evolution commands that adapt the relational database schema together with its instances, and update the contents of the meta-schema accordingly.

Conversely, in (Kaas, Pedersen, & Rasmussen, 2004) the evolution problem is investigated with particular reference to its impact on the logical level for ROLAP implementations, namely, on star and snowflake schemata. Eight basic evolution operators are defined (insert/delete dimension, level, property, and measure). For each of them, the changes implied on star and snowflake schemata are described and their impact on existing SQL queries in reporting tools is discussed. Remarkably, an in-depth comparison reveals that the star schema is generally more robust than the snowflake schema against schema changes.

A comprehensive approach to evolution is the one jointly devised at the Universities of Toronto and Buenos Aires. The fundamentals are laid by Hurtado, Mendelzon, & Vaisman (1999a), who propose a formal model for updating dimensions at both the schema and instance level, based on a set of modification operators (generalize, specialize, relate/unrelated/delete level are those defined at the schema level). An incremental algorithm for efficiently maintaining a set of materialized views in the presence of dimension updates is also presented. This work is then extended by Vaisman, Mendelzon, Ruaro, & Cymerman (2004) by introducing *TSOLAP*, an OLAP server supporting dimension updates and view maintenance, built following the OLE DB for OLAP proposal. The approach is completed by *MDDLX*, an extension of MDX

(Microsoft's language for OLAP) with a set of statements supporting dimension update operators at both schema and instance levels.

A relevant aspect related to evolution is how changes in schema affect the data mart quality, which is discussed in (Quix, 1999). A set of schema evolution operators is adapted from those for object-oriented databases; for each operator, its impact on the quality factors (such as completeness, correctness, and consistency between the conceptual and logical schema) as emerged in the context of the *DWQ Project - Foundations of Data Warehouse Quality* (Jarke, Jeusfeld, Quix, & Vassiliadis, 1999) is discussed. The tracking of the history of changes and the consistency rules to enforce when a quality factor has to be re-evaluated due to evolution is supported by an ad hoc meta-model.

## Versioning

According to the frequently cited definition by Inmon (1996), one of the characteristic features of a data warehouse is its non-volatility, which means that data is integrated into the data warehousing system once and remains unchanged afterwards. Importantly, this feature implies that the re-execution of a single query will always produce the same result. In other words, past analysis results can be verified and then inspected by means of more detailed OLAP sessions at any point in time. While non-volatility in the presence of changes at the data level can be achieved by adopting one of the solutions discussed in the third section, non-volatility in the presence of changes at the schema level requires some versioning approach to be undertaken. In fact, it is easy to see that the ability to re-execute previous queries in the presence of schema changes requires access to past schema versions, which cannot be achieved with an evolution approach.

The first work in this direction is COMET (Eder, Koncilia, & Morzy, 2002), a metamodel that supports schema and instance versioning. All classes in the metamodel are timestamped with a validity interval, so multiple, subsequent

versions of cubes can be stored and queried. Transformation of data from one version into the (immediate) succeeding or preceding one is supported; though the paper reports no details on how a new version can be obtained from the previous one, a comprehensive set of constraints that the versions have to fulfill in order to ensure the integrity of the temporal model is proposed.

The peculiarity of the timestamp-based versioning model proposed by Body, Miquel, Bédard, and Tchounikine (2003) is that hierarchies are deduced from the dimensions instances, so that explicitly defining the multidimensional schema is not necessary. In this way, schema changes are implicitly managed as a result of handling changes in instances. On the other hand, the versioning approach proposed by Ravat, Teste, & Zurfluh (2006) uses a constellation of star schemata to model different versions of the same fact, and populates versions by means of mapping functions.

A comprehensive approach to versioning is presented by Wrembel and Bebel (2007). Essentially, they propose two metamodels: one for managing a multi-version data mart and one for detecting changes in the operational sources. A multi-version data mart is a sequence of versions, each composed of a schema version and an instance version. Remarkably, besides “real” versions determined by changes in the application domain or in users’ requirements, also “alternative” versions are introduced, to be used for simulating and managing hypothetical business scenarios within what-if analysis settings.

Another approach to versioning specifically oriented to supporting cross-version queries is the one by Golfarelli, Lechtenbörger, Rizzi and Vossen (2006). Here, multidimensional schemata are represented as graphs of simple functional dependencies, and an algebra of graph operations to define new versions is defined. Data migration from the old to the new version is semi-automated, i.e., based on the differences between the two versions the system suggests a set of migration actions and gives support for their execution. The key idea of this approach

is to support flexible cross-version querying by allowing the designer to enrich previous versions using the knowledge of current schema modifications. For this purpose, when creating a new schema version the designer may choose to create *augmented schemata* that extend previous schema versions to reflect the current schema extension, both at the schema and the instance level. In a nutshell, the augmented schema associated with a version is the most general schema describing the data that are actually recorded for that version and thus are available for querying purposes. Like for migration, a set of possible augmentation actions is proposed to the designer (e.g., the designer may choose to manually insert values of a newly added attribute for hierarchy instances whose validity was limited to previous versions).

To the best of our knowledge, only two approaches use both valid and transaction time in the context of versioning. *Koncilia* (2003) presents a bi-temporal extension of the COMET metamodel, aimed at representing not only the valid time of schema modifications, but also the transaction time. Rechy-Ramírez and Benítez-Guerrero (2006) introduce a conceptual model for bi-temporal versioning of multidimensional schemata, aimed at enabling modifications in the data mart schema without affecting the existing applications. Each version has a temporal pertinence composed by a valid time and a transaction time, thus enabling the existence of two or more versions with the same valid time, but different transaction times. Associated to this model, there are 16 operators for schema changing and a SQL-like language to create and modify versions.

## QUERYING TEMPORAL DATA

The development of a model for temporal data warehousing is of little use without an appropriate query language capable of effectively handling time. In principle, a temporal query could be directly formulated on a relational schema using standard SQL, but this would

be exceedingly long and complex even for a skilled user.

In this direction, Bliujute, Saltenis, Slivinskias, & Jensen (1998) discuss the performance of their temporal star schema considering five types of temporal queries. Golfarelli & Rizzi (2007b) distinguish three querying scenarios in presence of late measurements:

- *Up-to-date queries*, that require the most recent measurement for each event;
- *Rollback queries*, that require a past version measurement for each event;
- *Historical queries*, that require multiple measurements for events, i.e., are aimed at reconstructing the history of event changes.

To cope with schema changes, Mendelzon and Vaisman (2000) proposed the *Temporal OLAP* (TOLAP) query language. TOLAP, based on the temporal multidimensional model proposed by Hurtado et al. (1999b), fully supports schema evolution and versioning, differently from best-known temporal query languages such as TSQL2 (Snodgrass, 1995), that supports versioning in a limited way only. TOLAP combines the temporal features of TSQL2 with some high-order features of SchemaLog in order to support querying multidimensional data with reference to different instants in time in a concise and elegant way. All three querying scenarios (today is yesterday, yesterday is today, and today or yesterday) are supported. Also meta-queries, e.g. concerning the instant changes to data took place, can be expressed.

Several approaches face the problem of formulating cross-version querying, i.e., formulating queries that span different schema versions. For instance, Morzy and Wrembel (2004) propose a SQL extension aimed at expressing queries on multiple (either real or alternative) schema versions. Each query is decomposed into a set of partial queries, one for each schema version involved. The results of partial queries are separately presented, annotated with version and metadata information; in some cases, partial queries results can be

merged into a common set of data. In (Wrembel & Bebel, 2007), the problem of cross-version queries is addressed by allowing users to specify either implicitly (by specifying a time interval for the query) or explicitly (by specifying a set of version identifiers) the set of versions for querying. Similarly, in (Golfarelli & Rizzi, 2007a) the relevant versions for answering a query are either chosen explicitly by the user or implicitly by the system based on the time interval spanned by the query, as shown in the prototype implementation X-Time.

In the context of querying, a number of works are related to the so-called *temporal aggregation problem*, that was studied mainly in the context of MOLAP systems and consists in efficiently computing and maintaining temporal aggregates. In fact, time dimensions typically lead to a high degree of sparseness in traditional array-based MOLAP cubes because of their large cardinality, and to significant overhead to answer time-parameterized range queries. For instance, the work by Tao, Papadias, & Faloutsos (2004) focuses on approximate temporal aggregate processing. Specifically, for count queries, its goal is to provide answers guaranteed to deviate from the exact ones within a given threshold. Riedewald, Agrawal, & El Abbadi (2002) proposed efficient range aggregation in temporal data warehouses by exploiting the append-only property of the time-related dimension. Their framework allows large amounts of new data to be integrated into the warehouse and historical summaries to be efficiently generated, independently of the extent of the data set in the time dimension. Feng, Li, Agrawal, & El Abbadi (2005) proposed a general approach to improve the efficiency of range aggregate queries on MOLAP data cubes in a temporal data warehouse by separately handling time-related dimensions to take advantage of their monotonic trend over time. Finally, Yang & Widom (2001) introduce a new index structure called the SB-tree, which supports fast lookup of aggregate results based on time, and can be maintained efficiently when the data changes along the time line.

## DESIGNING TEMPORAL DATA WAREHOUSES

It is widely recognized that designing a data warehousing system requires techniques that are radically different from those normally adopted for designing operational databases (Golfarelli & Rizzi, 1999). On the other hand, though the literature reports several attempts to devise design methodologies for data warehouses, very few attention has been posed on the specific design issues related to time. Indeed, as stated by Rizzi et al. (2006), devising design techniques capable of taking time and changes into account is one of the open issues in data warehouse research.

Pedersen and Jensen (1999) recognize that properly handling time and changes is a must-have for multidimensional models. Sarda (1999) summarizes the distinguishing characteristics of time dimensions: they are continuously valued and constantly increasing, they can be associated with multiple user-defined calendars, they express the validity of both facts and other dimensions (either in the form of time instants or validity intervals). Sarda also proposes a design methodology for temporal data warehouses featuring two phases: logical design, that produces relations characterized by a temporal validity, and physical design, that addresses efficient storage and access.

Considering the leading role played by temporal hierarchies within data marts and OLAP queries, it is worth adopting ad hoc approaches for their modeling not only from the logical, but also from the conceptual point of view. While all conceptual models for data marts allow for temporal hierarchies to be represented like any other hierarchies, to the best of our knowledge the only approach that provides ad hoc concepts for modeling time is the one by Malinowski & Zimányi (2008), based on a temporal extension of the MultiDim conceptual model. Different temporality types are allowed (namely, valid time, transaction time, lifespan, and loading time), and temporal support for levels, properties, hierarchies, and measures is granted.

Finally, Golfarelli & Rizzi (2007b) discuss the different design solutions that can be adopted in presence of late measurements, depending on the flow or stock nature of the events and on the types of queries to be executed.

## OPEN ISSUES AND CONCLUSIONS

In this survey we classified and discussed the issues related to temporal data warehousing. An in-depth analysis of the literature revealed that the research community not always devoted a comprehensive attention to all these aspects. As a matter of fact, a wide agreement on the possible design solutions has been reached only with reference to changes in dimensional data. As to changes in factual data and changes in schema, though some interesting solutions have been proposed, no broad and shared framework has been devised yet.

Similarly, on the commercial side, changes in data have been supported since almost a decade ago. Already in year 2000, systems such as Business Warehouse by SAP (2000) were allowing to track changes in data and to effectively query cubes based on different temporal scenarios by letting users choose which version of the hierarchies to adopt for querying. On the other hand, today there still is very marginal support to changes in schema by commercial tools. For instance, *SQL Compare* compares and synchronizes SQL Server database schemata, and can be used when changes made to the schema of a local database need to be pushed to a central database on a remote server. Also, the *Oracle Change Management Pack* is aimed to report and track the evolving state of meta-data, thus allowing to compare database schemata, and to generate and execute scripts to carry out the changes. In both cases, formulating a single query spanning multiple databases with different schemata is not possible.

We believe that, considering the maturity of the field and the wide diffusion of data warehousing systems, in the near future decision makers will be more and more demanding for

advanced temporal support. Thus, it is essential that both vendors and researchers be ready to deliver effective solutions. In this direction we envision two main open issues. On the one hand, some research aspects indeed require further investigation. For instance, support for cross-version queries is not satisfactory yet, and its impact on performance has not been completely investigated; similarly, the effectiveness of view adaptation approaches is still limited. On the other hand, in order to encourage vendors to add full temporal support to commercial platforms, the solutions proposed in the literature should be better harmonized to converge into a complete, flexible approach that could be effortlessly accepted by the market.

## REFERENCES

- Abelló, A., & Martín, C. (2003). A Bi-temporal Storage Structure for a Corporate Data Warehouse. *Proceedings International Conference on Enterprise Information Systems*, Angers, France, 177-183.
- Bellahsene, Z. (1998). View Adaptation in Data Warehousing Systems. *Proceedings International Conference on Database and Expert Systems Applications*, Vienna, Austria, 300-309.
- Bellahsene, Z. (2002). Schema Evolution in Data Warehouses. *Knowledge and Information Systems*, 4(3), 283-304.
- Blaschka, M. (2000). *FIESTA - A Framework for Schema Evolution in Multidimensional Databases*. PhD Thesis, Technische Universität München, Germany.
- Blaschka, M., Sapia, C., & Höfling, G. (1999). On Schema Evolution in Multidimensional Databases. *Proceedings International Conference on Data Warehousing and Knowledge Discovery*, Florence, Italy, 153-164.
- Bluijute, R., Saltenis, S., Slivinskas, G., & Jensen, C. S. (1998). Systematic Change Management in Dimensional Data Warehousing. *Proceedings International Baltic Workshop on Databases and Information Systems*, Riga, Latvia, 27-41.
- Body, M., Miquel, M., Bédard, Y., & Tchounikine, A. (2003). Handling Evolutions in Multidimensional Structures. *Proceedings International Conference on Data Engineering*, Bangalore, India, 581-591.
- Bruckner, R., & Tjoa, A. (2002). Capturing Delays and Valid Times in Data Warehouses - Towards Timely Consistent Analyses. *Journal of Intelligent Information Systems*, 19(2), 169-190.
- Chamoni, P. & Stock, S. (1999). Temporal Structures in Data Warehousing. *Proceedings International Conference on Data Warehousing and Knowledge Discovery*, Florence, Italy, 353-358.
- Chen, S., Zhang, X., & Rundensteiner, E. (2006). A Compensation-Based Approach for View Maintenance in Distributed Environments. *IEEE Transactions of Knowledge and Data Engineering*, 18(8), 1068-1081.
- Combi, C. & Oliboni, B. (2007). Temporal semistructured data models and data warehouses. In *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, Wrembel & Koncilia (Eds.), IRM Press, 277-297.
- De Amo, S., & Halfeld Ferrari Alves, M. (2000). Efficient Maintenance of Temporal Data Warehouses. *Proceedings International Database Engineering and Applications Symposium*, Yokohoma, Japan 188-196.
- Devlin, B. (1997). Managing Time In The Data Warehouse. *InfoDB*, 11(1), 7-12.
- Eder, J., & Koncilia C. (2001). Changes of Dimension Data in Temporal Data Warehouses. *Proceedings International Conference on Data Warehousing and Knowledge Discovery*, Munich, Germany, 284-293.
- Eder, J., Koncilia, C., & Morzy, T. (2002). The COMET Metamodel For Temporal Data Warehouses. *Proceedings International Conference on Advanced Information Systems Engineering*, Toronto, Canada, 83-99.
- Fan, H., & Poullovassilis, A. (2004). Schema Evolution in Data Warehousing Environments - A Schema Transformation-Based Approach. *Proceedings International Conference on Conceptual Modeling*, Shanghai, China, 639-653.
- Feng, Y., Li, H.-G., Agrawal, D., & El Abbadi, A. (2005). Exploiting Temporal Correlation in Temporal Data Warehouses. *Proceedings International Conference on Database Systems for Advanced Applications*, Beijing, China, 662-674.



- Golfarelli, M. & Rizzi, S. (1999). Designing the data warehouse: key steps and crucial issues. *Journal of Computer Science and Information Management*, 2(1), 1-14.
- Golfarelli, M., Lechtenbörger, J. Rizzi, S., & Vossen, G. (2006). Schema Versioning in Data Warehouses: Enabling Cross-Version Querying via Schema Augmentation. *Data and Knowledge Engineering*, 59(2), 435-459.
- Golfarelli, M. & Rizzi, S. (2007a). X-Time: Schema Versioning and Cross-Version Querying in Data Warehouses. *Proceedings International Conference on Data Engineering*, Istanbul, Turkey, 1471-147.
- Golfarelli, M. & Rizzi, S. (2007b). Managing late measurements in data warehouses. *International Journal of Data Warehousing and Mining*, 3(4), 51-67.
- Grandi, F. (2002). A Relational Multi-Schema Data Model and Query Language for full Support of Schema Versioning. *Proceedings SEBD*, Portoferraio, Italy, 323-336.
- Gupta, A., & Mumick, I. S. (1995). Maintenance of materialized views: problems, techniques, and applications. *Data Engineering Bulletin*, 18(2), 3-18.
- Hurtado, C., Mendelzon, A., & Vaisman, A. (1999a). Maintaining Data Cubes under Dimension Updates. *Proceedings International Conference on Data Engineering*, Sydney, Australia, 346-355.
- Hurtado, C., Mendelzon, A., & Vaisman, A. (1999b). Updating OLAP Dimensions. *Proceedings International Workshop on Data Warehousing and OLAP*, Kansas City, USA, 60-66.
- Inmon, W. (1996). *Building the data warehouse*. John Wiley & Sons.
- Jarke, M., Jeusfeld, M., Quix, C., & Vassiliadis, P. (1999). Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Information Systems*, 24(3), 229-253.
- Jensen, C., Clifford, J., Elmasri, R., Gadia, S. K., Hayes, P. J., & Jajodia, S. (1994). A Consensus Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 23(1), 52-64.
- Kaas, C., Pedersen, T. B., & Rasmussen, B. (2004). Schema Evolution for Stars and Snowflakes. *Proceedings International Conference on Enterprise Information Systems*, Porto, Portugal, 425-433.
- Kimball, R. (1996). *The Data Warehouse Toolkit*. Wiley Computer Publishing.
- Kimball, R. (2000). Backward in Time. *Intelligent Enterprise Magazine*, 3(15).
- Koncilia, C. (2003). ABi-Temporal Data Warehouse Model. *Short Paper Proceedings Conference on Advanced Information Systems Engineering*, Klagenfurt/Velden, Austria.
- Lee, A., Nica, A., & Rundensteiner, E. (2002). The EVE Approach: View Synchronization in Dynamic Distributed Environments. *IEEE Transactions on Knowledge and Data Engineering*, 14(5), 931-954.
- Lenz, H. J. & Shoshani, A. (1997). Summarizability in OLAP and Statistical Databases. *Proceedings Statistical and Scientific Database Management Conference*, Olympia, US, 132-143.
- Letz, C., Henn, E., & Vossen, G. (2002). Consistency in Data Warehouse Dimensions. *Proceedings International Database Engineering and Application Symposium*, Edmonton, Canada, 224-232.
- Malinowski, E. & Zimányi, E. (2008). A conceptual model for temporal data warehouses and its transformation to the ER and the object-relational models. *Data & Knowledge Engineering*, 64, 101-133.
- McKenzie, E., & Snodgrass, R. (1990). Schema Evolution and the Relational Algebra. *Information Systems*, 15(2), 207-232.
- Mendelzon, A., & Vaisman, A. (2000). Temporal queries in OLAP. *Proceedings Conference on Very Large Data Bases*, Cairo, Egypt, 242-253.
- Morzy, T. & Wrembel, R. (2004). On querying versions of multiversion data warehouse. *Proceedings International Workshop on Data Warehousing and OLAP*, Washington, DC, 92-101.
- Pedersen, T. B., & Jensen, C. (1998). Research Issues in Clinical Data Warehousing. *Proceedings Statistical and Scientific Database Management Conference*, Capri, Italy, 43-52.
- Pedersen, T. B. & Jensen, C. (1999). Multidimensional Data Modeling for Complex Data. *Proceedings International Conference on Data Engineering*, Sydney, Australia, 336-345.
- Quix, C. (1999). Repository Support for Data Warehouse Evolution. *Proceedings International*

*Workshop on Design and Management of Data Warehouses*, Heidelberg, Germany.

Ravat, F., Teste, O., & Zurfluh, G. (2006). A Multiversion-Based Multidimensional Model. *Proceedings International Conference on Data Warehousing and Knowledge Discovery*, 65-74.

Rechy-Ramírez, E.-J. & Benítez-Guerrero, E. (2006). A Model and Language for Bi-temporal Schema Versioning in Data Warehouses. *Proceedings International Conference on Computing*, Mexico City, Mexico.

Riedewald, M., Agrawal, D. & El Abbadi, A. (2002). Efficient integration and aggregation of historical information. *Proceedings SIGMOD Conference*, Madison, Wisconsin, 13-24.

Rizzi, S., Abelló, A., Lechtenböcker, J., & Trujillo, J. (2006). Research in Data Warehouse Modeling and Design: Dead or Alive? *Proceedings International Workshop on Data Warehousing and OLAP*, Arlington, USA, 3-10.

Roddick, J. (1995). A Survey of Schema Versioning Issues for Database Systems. *Information and Software Technology*, 37(7), 383-393.

SAP Institute (2000). *Multi-dimensional Modeling with SAP BW*. SAP America Inc. and SAP AG.

Sarda, N. L. (1999). Temporal Issues in Data Warehouse Systems. *Proceedings International Symposium on Database Applications in Non-Traditional Environments*, Kyoto, Japan, 27-34.

Tansel, A. U., Clifford, J., Gadia, S. K., Jajodia, S., Segev, A., & Snodgrass, R. T. (1993). *Temporal databases: theory, design and implementation*. Benjamin Cummings.

Snodgrass, R. T. (1995). *The SQL2 Temporal Query Language*. Kluwer Academic Publishers.

Tao, Y., Papadias, D., & Faloutsos, C. (2004). Approximate Temporal Aggregation. *Proceedings International Conference on Data Engineering*, Boston, Massachusetts, 190-201.

Vaisman, A., Mendelzon, A., Ruaro, W., & Cymerman, S. (2004). Supporting Dimension Updates in an OLAP Server. *Information Systems*, 29, 165-185.

Vaisman, A., & Mendelzon, A. (2001). A Temporal Query Language for OLAP: Implementation and a Case Study. *Proceedings DBPL*.

Wrembel, R. & Bebel, B. (2007). Metadata Management in a Multiversion Data Warehouse. *Journal of Data Semantics*, 8, 118-157.

Yang, J. & Widom, J. (1998). Maintaining Temporal Views over Non-Temporal Information Sources for Data Warehousing. *Proceedings International Conference on Extending Database Technology*, Valencia, Spain, 389-403.

Yang, J. & Widom, J. (2001). Incremental Computation and Maintenance of Temporal Aggregates. *Proceedings International Conference on Data Engineering*, Heidelberg, Germany, 51-60.

*Matteo Golfarelli received his PhD for his work on autonomous agents in 1998. In 2000 he joined the University of Bologna as a researcher. Since 2005 he is associate professor, teaching information systems and database systems. He has published over 60 papers in refereed journals and international conferences in the fields of data warehousing, pattern recognition, mobile robotics, multi-agent systems. He served in the PC of several international conferences and as a reviewer in journals. His current research interests include all the aspects related to business intelligence and data warehousing, in particular multidimensional modeling, what-if analysis and BPM.*

*Stefano Rizzi received his PhD in 1996 from the University of Bologna, Italy. Since 2005 he is full professor at the University of Bologna, where he is the head of the Data Warehousing Laboratory. He has published about 100 papers in refereed journals and international conferences mainly in the fields of data warehousing, pattern recognition, and mobile robotics. He joined several research projects on the above areas and has been involved in the PANDA thematic network of the European Union concerning pattern-*



*base management systems. His current research interests include data warehouse design and business intelligence, in particular multidimensional modeling, data warehouse evolution, OLAP preferences and what-if analysis.*

# Lossless Reduction of Datacubes using Partitions

*Alain Casali, Aix-Marseille Universités, France*

*Sébastien Nedjar, Aix-Marseille Universités, France*

*Rosine Cicchetti, Aix-Marseille Universités, France*

*Lotfi Lakhal, Aix-Marseille Universités, France*

*Noël Novelli, Aix-Marseille Universités, France*

---

## ABSTRACT

*Datacubes are especially useful for answering efficiently queries on data warehouses. Nevertheless the amount of generated aggregated data is huge with respect to the initial data which is itself very large. Recent research has addressed the issue of a summary of Datacubes in order to reduce their size. The approach presented in this paper fits in a similar trend. We propose a concise representation, called Partition Cube, based on the concept of partition and we give a new algorithm to compute it. We propose a Relational Partition Cube, a novel ROLAP cubing solution for managing Partition Cubes using the relational technology. Analytical evaluations show that the storage space of Partition Cubes is smaller than Datacubes. In order to confirm analytical comparison, experiments are performed in order to compare our approach with Datacubes and with two of the best reduction methods, the Quotient Cube and the Closed Cube.*

*Keywords:* concept lattices; datacubes; OLAP; partitions

---

## INTRODUCTION

In order to efficiently answer OLAP queries (Chaudhuri & Dayal, 1997), a widely adopted solution is to compute and materialize Datacubes (Gray et al., 1997). For example, given a relation  $r$  over the schema  $R$ , a set of dimensions  $\text{Dim} = \{C_1, C_2, C_3\}$ ,  $\text{Dim} \subseteq R$ , a measure  $M \in R$ , an aggregate function  $f$ , the cube operator (Gray et al., 1997) is expressed as follows:

```
SELECT  $C_1, C_2, C_3, f(M)$ 
FROM  $r$ 
CUBE BY  $(C_1, C_2, C_3)$ 
```

Dimensions are also called categorical attributes and  $r$  a categorical database relation. The given query achieves all the possible group-by according to any attribute combination belonging to the power set of  $\text{Dim}$ . It results in what is called a Datacube, and each sub-query

performing a single group-by yields a cuboid. Computing Datacubes is exponential in the number of dimensions (the lattice of the dimension set must be explored), and the problem worsens when very large data sets are to be aggregated. Datacubes are considerably larger than the input relation. (Ross & Srivastava, 1997) exemplifies the problem by achieving a full Datacube encompassing more than 210 millions of tuples from an input relation having 1 million of tuples. The problem is originated by a twofold reason: on one hand the exponential number of dimensional combinations to be dealt, and on the other hand the cardinality of dimensions. The larger dimension domains are, the more aggregated results there are (according to each real value combination). Unfortunately, it is widely recognized that in OLAP databases, data can be very sparse (Ross & Srivastava, 1997; Beyer & Ramakrishnan, 1999) thus scarce value combinations are likely to be numerous and, when computing entirely the Datacubes (full Datacubes), each exception must be preserved. In such a context, (1) approaches favour the efficiency of OLAP queries to the detriment of storage space or (2) they favour an optimal representation of cubes but OLAP query performances are likely to be debased (Kotidis & Roussopoulos, 1998).

## RELATED WORK

The approaches addressing the issue of Datacube computation and storage attempt to reduce at least one of the quoted drawbacks. The algorithms Buc (Beyer & Ramakrishnan, 1999) and Hcubing (Han, Pei, Dong, & Wang, 2001) enforce antimonotone constraints and partially compute Datacubes (iceberg cubes) to reduce both execution time and disk storage requirements. The underlying argument is that OLAP users are only interested in general trends (and not in atypical behaviors). With a similar argumentation, other methods use the statistic structure of data to compute density distributions

and give approximate answers to OLAP queries (Pedersen, Jensen, & Dyreson, 1999; Vitter & Wang, 1999; Shanmugasundaram, Fayyad, & Bradley, 1999; Gilbert, Kotidis, Muthukrishnan, & Strauss, 2001).

The above mentioned approaches are efficient and meet their twofold objective (reduction of execution time and space storage). However, they are not able to answer whatever query (although OLAP queries are, by their very nature, ad hoc queries (Han & Kamber, 2001)).

Another category of approaches is the so-called "information lossless". They aim to find the best compromise between OLAP query efficiency and storage requirements without discarding any possible query (even unfrequent). Their main idea (see for details (Harinarayan, Rajaraman, & Ullman, 1996; Kotidis & Roussopoulos, 1999; Sellis, 2004; Theodoratos & Xu, 2004; Gupta & Mumick, 2005)) is to pre-compute and store frequently used aggregates while preserving all the data (possibly at various aggregation levels) needed to compute on line the result of a not foreseen query. They are mostly found in view materialization research.

The following five methods<sup>1</sup> also fit in the information lossless trend: the Dwarf Cube (Sismanis, Deligiannakis, Roussopoulos, & Kotidis, 2002), the Condensed Cube (Wang, Lu, Feng, & Yu, 2002), the CURE for Cubes (Morfonios & Ioannidis, 2006), the Quotient Cube (L. Lakshmanan, Pei, & Han, 2002; L. V. S. Lakshmanan, Pei, & Zhao, 2003) and the Closed Cube (Casali, Cicchetti, & Lakhal, 2003b; Li & Wang, 2005; Xin, Shao, Han, & Liu, 2006). They favor the optimization of storage space while preserving the capability to answer whatever query. The two latter compute the two smallest representations of a Datacube and thus are the most efficient for both saving storage space and answering queries like "Is this behavior frequent or not?". From these two representations, the exact data of a whole Datacube can be retrieved by performing a computation on line, because results of queries are not precomputed and preserved.

## Contribution

In this paper we pursue a new research path while remaining in the trend of information lossless approaches. We propose a new representation which fully and exactly captures all the information enclosed in a Datacube. Unlike the other information lossless methods, our approach can answer any query without additional execution time. Moreover, our representation provides a simple mechanism reducing significantly the size of aggregates to be stored. More precisely, the contributions described in this paper are the following:

- we investigate the construction of Quotient Cube and Closed Cube. The choice of these two structures is motivated by their small size (Xin et al., 2006) and theoretical foundation: lattices and closure systems (Ganter & Wille, 1999). We show that it is possible to use existing efficient algorithms originally intended for extracting frequent patterns, in order to compute the quoted cube representations;
- we propose a new concise representation of Datacubes: the Partition Cube, based on simple concepts which extend the ones of the partitional model (Laurent & Spyrtatos, 1988). The concept of concise representation has been firstly introduced in (Mannila & Toivonen, 1996) for frequent itemsets and it is used in this paper as a reduced representation of Datacubes;
- we introduce a depth-first search algorithm called Pcube in order to build up the Partition Cube. Pcube enumerates the aggregates to be computed according to the lexic order (Ganter & Wille, 1999). We show that Pcube minimizes main memory requirements (and avoids swaps);
- by considering the most used environment when managing data warehouses (ROLAP), we propose a relational implementation of our solution (which can be easily achieved).
- finally, a detailed analytical and experimental comparison is made between our

representation and the Datacube, the Quotient Cube and the Closed Cube. We show that our representation provides an important reduction of storage space when compared to the Datacube. Meanwhile, as expected the Quotient Cube and the Closed Cube are smaller than the Partition Cube (even if, in theory and for extreme cases, the two former can be equal to the size Partition Cube when  $|\text{Dim}| > 2$ ). As an additional benefit of our method, once the Relational Partition Cube is stored, any query can be answered on line with no additional computation time.

The remainder of this article is organized as follows. The following Section focuses on the two approaches chosen as references. Our proposal is detailed in Section Partition Cubes. We define the concepts of our representation, the algorithm Pcube along with a relational implementation of our representation. We relate analytical and experimental evaluations in the last Sections. In conclusion, we resume the strengths of our contribution. This article consolidates and extends research presented in the international conference paper (Casali, Cicchetti, Lakhal, & Novelli, 2006).

## Lossless Reduction of Datacubes using Lattices

We focus, in this section, on the two most concise representations of the Datacubes which are information lossless and based on lattices: the Quotient Cube and the Closed Cube. They fit in the cube lattice framework of a categorical database relation  $r$ :  $\text{CL}(r)$  (Casali, Cicchetti, & Lakhal, 2003a). This framework is summarized in appendix and an outline is also presented in (Kudryavcev, 2006). The cube lattice is a search space (denoted by  $\text{space}(r)$ ) organizing the tuples, possible solutions of the problem, according to a generalization / specialization order, denoted by  $\preceq$ , capturing a similar semantics than Roll-Up/Drill-Down (Gray et al., 1997). These tuples share the same structure than the tuples of  $r$  but attributes dimensions

can be provided with the value ALL (Gray et al., 1997). Moreover, we append to these tuples a virtual tuple which only encompasses empty values in order to close the structure. Any tuple of the cube lattice generalizes the tuple of empty values. For handling the tuples of  $CL(r)$ , the operators Sum (+) and Product (•) are defined. Provided with a couple of tuples, the sum yields the most specific tuple which generalizes the two operands. The Product yields the most general tuple which specializes the two operands. After the description of the Quotient Cube and Closed Cube we show how to compute them using existing closed set algorithms.

## Quotient Cubes

A Quotient Cube is a summary of Datacube for aggregate functions like Count, Min, Max, Avg and Top-k (L. Lakshmanan et al., 2002). Moreover, the Quotient Cube preserves the semantics of the cube operators Rollup and Drill-down. When introducing the Quotient Cube, the authors use the concept of convex classes (i.e. sets of tuples). A class  $C$  is convex if and only if  $\forall c, c' \in C$ , if it exists  $c''$  such that  $c \preceq c'' \preceq c'$  thus  $c'' \in C$ . Classes are built as follows. Given an aggregate function  $f$ , the equivalence relation  $\equiv_f$  is defined as a transitive and reflexive closure of the following relation  $R$ : let  $t, t'$  be two tuples,  $tRt'$  holds if and only if (i)  $f(t) = f(t')$  and (ii)  $t$  is either a parent or a child of  $t'$ . Let us note by  $[t]_{\equiv_f}$  the equivalence class of  $t$  ( $[t]_{\equiv_f} = \{t' \in CL(r) \text{ such that } tRt'\}$ ). Moreover, the equivalence relation  $\equiv_f$  must satisfy the weak congruence property which can be expressed as follows:  $\forall c, c', d, d' \in CL(r)$ , if  $c \equiv_f c', d \equiv_f d', c \preceq d$  and  $d' \preceq c'$ , thus  $c \equiv_f d$  also holds. Weak congruence property implies the convex property for each equivalence class.

**Definition 1. (Quotient Cube Lattice)** - Let  $CL(r)$  be the cube lattice of the relation  $r$  and  $\equiv_f$  an equivalence relation on  $Space(r)$  satisfying the weak congruence property. The Quotient Cube Lattice  $QCL(r, f) = \langle \{[t]_{\equiv_f} f(t) \mid t \in CL(r)\}, \preceq_Q \rangle$  encompasses equivalence classes

induced by  $\equiv_f$ . For two equivalence classes  $A$  and  $B \in QCL(r, f)$ , we have  $A \preceq_Q B$  if  $\exists a \in A$  and  $\exists b \in B$  such that  $a \preceq b$ .

**Example 1.** Let us consider the Sale relation  $r$  (cf. table 1) which contains the attributes City ( $C$ ), Day ( $D$ ), Product ( $P$ ) and Quantity ( $Q$ ) yielding the quantity of products sold at a city for a given day. The Quotient Cube lattice of the relation in table 1 for the aggregate function Sum is illustrated in figure 1. Classes are provided with their maximal tuple, minimal tuples (w.r.t.  $\preceq$ ) and the value (in bold) of the measure (original or aggregated using the function Sum). If the maximal and minimal tuples are equal, the class is reduced to this single tuple.

## Closed Cubes

In formal concept analysis (Ganter & Wille, 1999), the closure operator (the intersection) groups together all the binary patterns sharing a similar support (Pasquier, Bastide, Taouil, & Lakhal, 1999). By adopting this principle, we define a closure operator on the cube lattice. This operator is denoted by  $\mathbb{C}$ . For any tuple  $t$  belonging to the search space,  $\mathbb{C}(t)$  yields the Sum of the tuples of the relation which are more specific than  $t$  (i.e:  $\mathbb{C}(t) = +t' : t' \in r \text{ and } t \preceq t'$ ). If for a tuple  $t$ , we have  $\mathbb{C}(t) = t$ , then we say that the tuple  $t$  is a closed tuple. We group the set of closed tuples in the cube closure system, denoted by  $\mathbb{C}(r)$  ( $\mathbb{C}(r) = \{\mathbb{C}(t) : t \in CL(r)\}$ ). Finally, we use Birkhoff theorem (Ganter & Wille, 1999) to construct the Closed Cube (Casali et al., 2003b).

**Theorem 1.** (Casali et al., 2003b). The poset  $CCL(r) = \langle \{(t, f(t)) \text{ such that } t \in \mathbb{C}(r)\}, \preceq \rangle$  is a complete and coatomistic lattice called the Closed Cube. Moreover, we have:

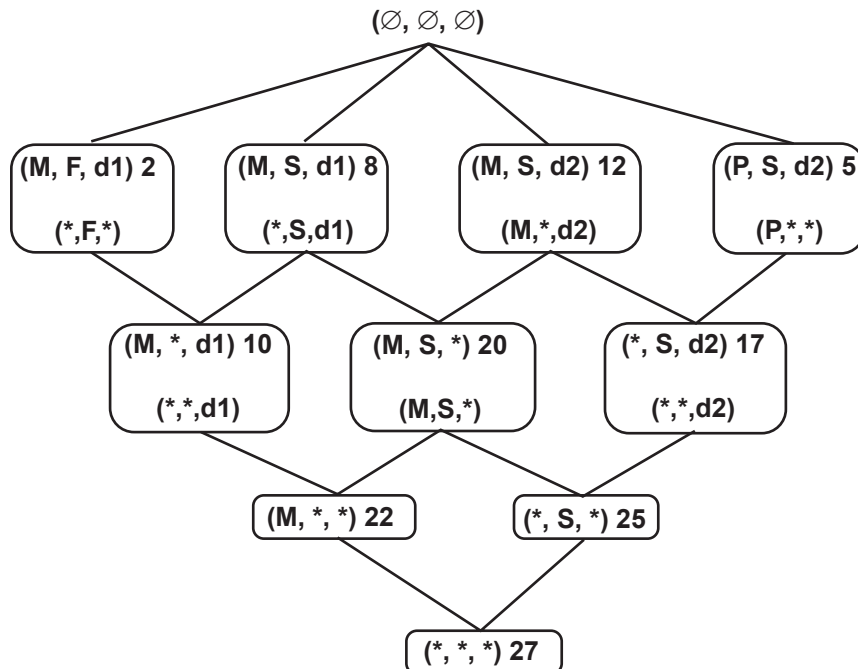
$$\forall t \subseteq CCL(r), \wedge T = +_{t \in T} t,$$

$$\forall t \subseteq CCL(r), \vee T = C(\bullet_{t \in T} t)$$

Where  $\wedge$  stands for the meet operator and  $\vee$  for the join operator (Ganter & Wille, 1999).

Table 1. Sale Relation

RowID	City	Product	Day	Quantity
1	Marseilles	Flower	d <sub>1</sub>	2
2	Paris	Sweet	d <sub>2</sub>	5
3	Marseilles	Sweet	d <sub>1</sub>	8
4	Marseilles	Sweet	d <sub>2</sub>	12

Figure 1. Quotient Cube lattice for the aggregate function Sum ( $*$   $\Leftrightarrow$  ALL)

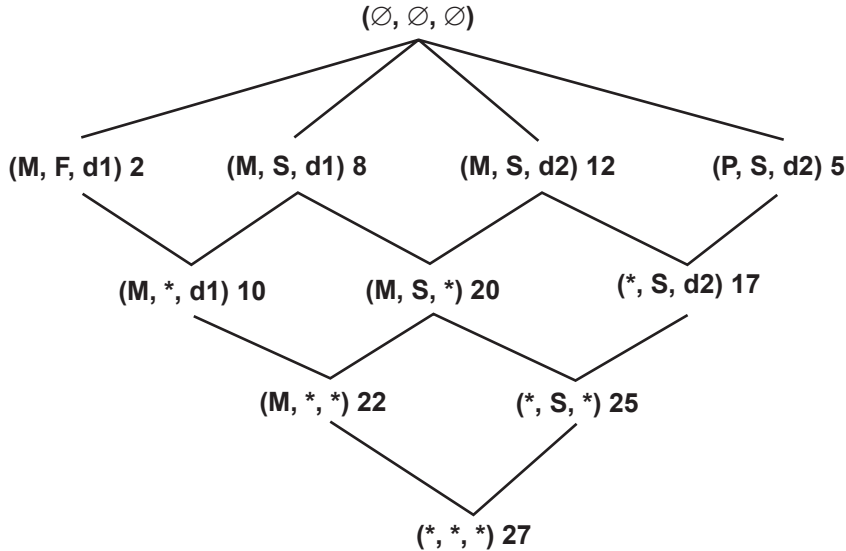
**Example 2.** Figure 2 illustrates the Closed Cube of the relation  $r$  which is isomorphic to the Quotient Cube (cf. figure 1).

### How to Compute Closed and Quotient Cubes

The Closed Cube is isomorphic to the Quotient Cube for the aggregate functions Count and Sum (if all the values of the attribute measure are strictly positive or negative). We can construct a binary relation associated to the database relation  $r$  by applying an embedded order (Casali et al., 2003a) over each tuple of  $r$ . This embedded

order associates to any value of any attribute a single item in the associated binary relation. There is a lattice isomorphism between the Closed Cube and the Galois lattice (concept lattice) computed from the underlying binary relation. We show below how provided with such an isomorphism it is possible to compute Closed Cubes by reusing existing algorithms.

**Theorem 2.** (Casali et al., 2003b). Let  $r$  be a categorical database relation. Then the Closed Cube is a concise representation of a Data-cube of  $r$  for the aggregate functions Count, Sum, Min, Max, Avg and Top- $k$ . Moreover:

Figure 2. Hasse diagram of the Closed Cube of  $r$ 

- the Closed Cube is isomorphic to the Quotient Cube for the aggregate functions Count and Sum (under the mentioned conditions on the measure values);
- the Closed Cube is isomorphic to the Galois lattice computed on the binary relation resulting by the application of the embedded order over each tuple of the database relation.

The consequences of theorem 2 are especially attractive since they make it possible the use of algorithms, proved to be very efficient in a binary context, in order to construct the Quotient Cube (e.g: Close (Pasquier et al., 1999) and Titanic (Stumme, Taouil, Bastide, Pasquier, & Lakhal, 2002)) or for computing the Closed Cube (Close (Pasquier et al., 1999), Titanic (Stumme et al., 2002), Charm (Zaki & Hsio, 2002), Closet (Pei, Han, & Mao, 2000)).

We use the algorithm Close (because it can compute both the Quotient Cube and the Closed Cube) to make experimental comparisons of our new proposal with the two approaches described in this section.

## Partition Cubes

The approach introduced in this paper proposes a concise representation of Datacubes: the Partition Cube. We start this section by presenting the concepts of our approach. Then we present an algorithm for computing such a representation. Finally, by considering the most used environment when managing data warehouses, we propose a relational implementation of our solution (which can be easily achieved).

## Basic Concepts

In this section, we introduce a new characterization of Datacube based on simple concepts using partitions. The following definition uses the concept of agree sets (Mannila & Toivonen, 1996; Lopes, Petit, & Lakhal, 2002).

**Definition 2 (DM-Class).** Let  $r$  be a categorical database relation and  $X$  a set of dimension attributes. A dimension-measure class (DM-Class), of a tuple  $t$  according to  $X$ ,  $[t]X$ , is defined by the set of couples (identifier( $u$ ), measure( $u$ )) of all the tuples  $u$  which agree with  $t$  according to a set of attributes  $X$  (i.e: the set of tuples  $u$



having the same values as  $t$  for  $X$ ). Thus, we have:  $[t]_X = \{(u[\text{RowId}], u[M]) \text{ such that } u[X] = t[X], \forall u \in r\}$ .

Each DM-Class is represented by a couple of numbers: the former is one of the identifiers of the original tuples gathered within the considered class, and the latter is the computed measure for the class.

**Example 3.** With our relation example, the DM-Class associated to the first tuple according to the dimension attribute City groups all the couples (identifier, measure) for tuples satisfying, like  $t_p$ , the constraint (City='Marseilles'):  $[t]_{\text{City}} = \{(1; 2); (3; 8); (4; 12)\}$ .

All the DM-Classes for a dimension attribute set  $X$  are gathered within a single set: the Dimension-Measure Partition (DM-Partition).

**Definition 3 (DM-Partition).** Let  $r$  be a categorical database relation and  $X$  a set of dimension attributes, the DM-Partition of  $r$  according to  $X$ , is defined as  $\Pi_X(r) = \{[t]_X \mid \forall t \in r\}$ .

**Example 4.** In our examples, for a better readability, the DM-Classes are delimited by the symbols '<' and '>' when writing the DM-Partitions. Thus, in our example, the DM-Partition associated to the attribute City is:  $\Pi_{\text{City}}(r) = \{<(1,2), (3,8), (4,12)>, <(2,5)>\}$ . The one associated to the attribute Product is:  $\Pi_{\text{Product}}(r) = \{<(1,2)>, <(2,5), (3,8), (4,12)>\}$ .

Let us consider two DM-Partitions computed according to the attribute sets  $X$  and  $Y$ . Their product yields the DM-Partition according to  $X \cup Y$ . Such a product is performed by intersecting DM-classes of the two DM-Partitions and preserving only not empty classes (cardinality greater than or equal to 1).

**Lemma 1. (Product of DM-Partitions).** Let  $r$  be a categorical database relation,  $X$  and  $Y$  two sets of dimension attributes,  $\Pi_X(r)$  and  $\Pi_Y(r)$  their DM-Partitions respectively. The product of the DM-Partitions  $\Pi_X(r)$  and  $\Pi_Y(r)$ , noted by  $\Pi_X(r) \bullet_p \Pi_Y(r)$ , returns the DM-Partition over  $X \cup Y$  and is obtained as follows:  $\Pi_X(r) \bullet_p$

$\Pi_Y(r) = \Pi_X \cup_Y(r) = \{[t]_Z = [t]_X \cap [t]_Y : [t]_Z \neq \emptyset, [t]_X \in \Pi_X(r) \text{ and } [t]_Y \in \Pi_Y(r)\}$ .

**Example 5.** Given our relation, the DM-Partitions related to the attributes City and Product are the following:  $\Pi_{\text{City}}(r) = \{<(1, 2), (3, 8), (4, 12)>, <(2, 5)>\}$  and  $\Pi_{\text{Product}}(r) = \{<(1, 2)>, <(2, 5), (3, 8), (4, 12)>\}$ . Thus  $\Pi_{\text{City, Product}}(r) = \Pi_{\text{City}}(r) \bullet_p \Pi_{\text{Product}}(r) = \{<(1, 2)>, <(2, 5)>, <(3, 8), (4, 12)>\}$ .

Once the DM-Partitions are computed, the cuboids of the Datacube can be easily obtained. Any DM-Class originates a tuple of a cuboid and the measure value is achieved by applying the aggregate function on the set of measure values of the DM-Class.

**Example 6.** Since we have  $\Pi_{\text{City, Product}}(r) = \{<(1, 2)>, <(2, 5)>, <(3, 8), (4, 12)>\}$ , thus the cuboid according to City, Product is composed of three tuples:

(M; F; ALL; 2), (T; S; ALL; 5) and (M; S; ALL; 20).

All the couples standing for the DM-Classes are grouped within a set: the Partition Cuboid.

**Definition 4 (Partition Cuboid).** Let  $\Pi_X(r)$  be a DM-Partition of  $r$  and  $f$  an aggregate function. For each DM-Class,  $[t]_X \cdot M$  is the value of the measure attribute. The Partition Cuboid according to the attribute set  $X$ , denoted by  $\text{Cuboid}_X(r)$ , is defined as follows:  $\text{Cuboid}_X(r) = \{(t[\text{RowId}], f([t]_X \cdot M)), \forall [t]_X \in \Pi_X(r)\}$ .

**Example 7.** The Partition Cuboid according to City; Product is the following:

$\text{Cuboid}_{\text{City}}(r) = \{(1; 2); (2; 5); (3; 20)\}$ .

Our representation of the Datacube can be defined as the whole set of Partition Cuboids according to any dimension combination.

**Definition 5 (Partition Cube).** Let  $r$  be a categorical database relation. The Partition Cube associated to  $r$  is defined as:  $\text{Partition\_Cube}(r)$

$= \{Cuboid_X(r), \forall X \in \wp(D)\}$ , where  $\wp$  stands for the powerset lattice.

**Example 8.** The Partition Cube for the aggregate function Sum is given in table 2. It contains  $2^3 = 8$  cuboids (because there are 3 dimensions), each of which corresponding to a dimension combination (used as an index to identify cuboids).

## The Pcube Algorithm

In this section, we describe the principles of our algorithmic solution. First, we give a simple definition of the lectic order (co-lexicographical order) (Ganter & Wille, 1999). Then, we propose a new recursive algorithm for enumerating, according to the lectic order, the subsets of  $\wp(D)$ .

**Definition 6 (Lectic Order).** Let  $(D, <_D)$  be a finite set totally ordered. We assume, by simplicity, that  $D$  can be defined as follows:  $D = \{A_1, A_2, \dots, A_n\}$ .  $D$  is provided with the following operator:

$Max : \wp(D) \rightarrow D$

$X \mapsto$  the last element of  $X$  according to  $<_D$ . The lectic order, denoted by  $<_r$ , is defined as follows:  $\forall X, Y \in \wp(D), X <_r Y, Max(X \setminus Y) < Max(Y \setminus X)$ . This order is a strict linear order over the set of all subsets of a set.

**Example 9.** Let us consider the following totally order set  $D = \{A, B, C, D\}$ . Enumerating the combinations of  $\wp(D)$ , with respect to the lectic order, provides the following results:  $\emptyset <_r A <_r B <_r AB <_r C <_r AC <_r BC <_r ABC <_r D <_r AD <_r BD <_r ABD <_r CD <_r ACD <_r BCD <_r ABCD$ .

**Proposition 1. (Ganter & Wille, 1999).**  $\forall X, Y \in \wp(D), X \subset Y \Leftrightarrow X <_r Y$ .

We firstly present the new algorithm Ls (Lectic Subsets) which gives the general algorithmic schema used by Pcube (the algorithm building the Partition Cube).

## Recursive Algorithmic Schema for Enumerating the Subsets in Llectic Order

The new algorithm LS has as parameters two dimensional attribute subsets  $X$  and  $Y$ . The algorithm is based on a twofold recursion. The recursive calls form a binary balanced tree in which each execution branch returns a dimensional subset. The general strategy for enumerating dimensional attribute combinations consists in considering firstly all the subsets not encompassing a dimensional attribute, and then all the subsets which encompass it. More precisely, the maximal attribute, according to the lectic order, is discarded from  $Y$  and added to  $X$  in the variable  $Z$ . The algorithm is recursively applied with (i)  $X$  and a new subset  $Y$  (from which the maximal attribute is pruned), then

Table 2. Partition Cube for the aggregate function Sum

Cuboid <sub><math>\emptyset</math></sub>	{(1, 27)}
Cuboid <sub>C</sub>	{(1, 22), (2, 5)}
Cuboid <sub>D</sub>	{(1, 10), (2, 17)}
Cuboid <sub>B</sub>	{(1, 2), (2, 25)}
Cuboid <sub>CD</sub>	{(1, 10), (2, 5), (4, 12)}
Cuboid <sub>CP</sub>	{(1, 2), (2, 5), (3, 20)}
Cuboid <sub>DP</sub>	{(1, 2), (2, 17), (3, 8)}
Cuboid <sub>CDP</sub>	{(1, 2), (2, 5), (3, 8), (4, 12)}

Algorithm 1. Algorithm LS

Input :  $X$  and  $Y$  two sets of dimensions  
Output :  $\wp(D)$

```

1.  if  $Y = \emptyset$  then
2.    Return  $X$ 
3.  Else
4.     $A := \max(Y)$ 
5.     $Y := Y \setminus \{A\}$ 
6.    LS( $X, Y$ )
7.     $Z := X \cup \{A\}$ 
8.    LS( $Z, Y$ )
9.  End if

```

(ii)  $Z$  and  $Y$ . The first call of  $Ls$  is provided with two parameters  $X = \emptyset$  and  $Y = D$ .

**Lemma 2.** *The correctness of the algorithm  $Ls$  is based on proposition 1 and the distributive property of the dimension attribute lattice. We have:  $\forall A \in D, \forall X \subseteq \wp(D), P(X \cup A) \cap \wp(D \setminus (X \cup A)) = \emptyset$ . Thus, each subset of dimension attributes is enumerated exactly once.*

**Example 10.** *Let us consider our relation  $r$ . In this context, the binary tree of recursive calls when running our algorithm is depicted in figure 3. The leaves in the tree correspond to outputs which are, from left to right, ordered in a lexic way. In any left subtree, all subsets not encompassing the maximal attribute (according to the lexic order) of the subtree root are considered while in right subtrees, the maximal attribute is preserved.*

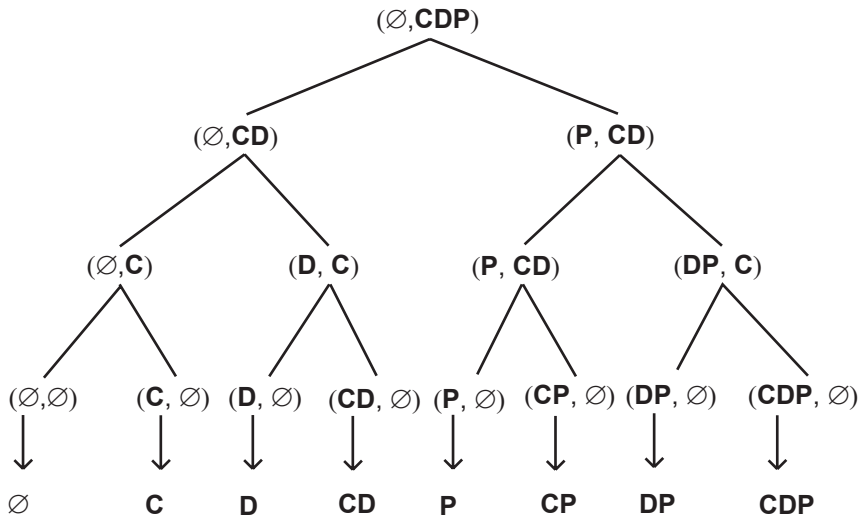
Now, we can introduce our algorithm, called  $Pcube$ , for computing Partition Data-cubes. As mentioned,  $Pcube$  fits in the theoretical framework previously presented. A pre-processing step is required in order to build DM-Partitions according to each single attribute from the input relation. Performing this

initial step also computes the empty set based cuboid ( $Cuboid_{\emptyset}$ ) and its result is yielded. If the original partitions ( $\bigcup_{A \in D} \prod_A(r)$ ) cannot fit in main memory, then the fragmentation strategy proposed in (Ross & Srivastava, 1997) and used in (Beyer & Ramakrishnan, 1999) is applied. Its main idea is to divide the input relation in fragments according to an attribute until the original associated DM-Partitions can be loaded.  $Pcube$  adopts the general algorithm schema of  $Ls$  but it is intended to compute all desired aggregates and thus it yields the condensed representation of all possible cuboids.  $Pcube$  deals with DM-partitions and enforces product of DM-partitions. Like  $Ls$ , its input parameters are the subsets of dimensions  $X$  and  $Y$ . The DM-Partition associated to  $Z$  is computed by applying the product over the two partitions in memory:  $\prod_X(r)$  and  $\prod_A(r)$ . The second recursive call is performed. The pseudo-code of the algorithm  $Pcube$  is given below.

## Relational Partition Cubes

When the OLAP application is managed by a relational system, the Partition Cube can be stored tuple describes a DM-Class of a cuboid

Figure 3. Execution tree of  $Ls$ .



**Algorithm 2. Algorithm Pcube**


---

Input : Set of DM-Partition  $\{\prod_A, A \in D\}$ ,  
X and Y two sets of dimensions attributes  
Output : Partition Cube

```

1.  if Y =  $\emptyset$  then
2.    Write_Cuboid(X)
3.  Else
4.    A := max (Y)
5.    Y := Y \ {A}
6.    PCUBE(r, X, Y)
7.     $\prod_Z := \prod_X \cdot \prod_A$ 
8.    PCUBE(r, Z, Y)
9.  End if

```

---

according to X. More precisely, for each DM-Class, are known the identifier of its representing element, the measure value and the dimension combination (DimId). Like in the other approaches computing cubes, real values of dimensions are encoded with integers (Ross & Srivastava, 1997; Beyer & Ramakrishnan, 1999; Han et al., 2001). We propose the following schema called relational Partition Cubes.

r(RowId, D, M)  
Dimension(DimId, D)  
Cube(RowId, DimId, f(M))

To compute Relational Partition Cubes from Datacubes using the algorithm LS, PL/SQL procedures can be downloaded at <http://infodoc.iut.univ-aix.fr/~casali/PL-RPC.zip>. The relation Dimension is intended for storing all the

dimension combinations. Its values are binary and for any attribute A, A has the value 0 if it does not belong to the considered combination, else its value is 1. Finally the original relation makes it possible to retrieve the real values of dimensions for various representing elements of the DM-Classes.

**Example 11.** For our example, the two latter relations of the schema implementing our concise representation are given in table 3 and 4 respectively.

**Analytical Evaluation**

When computing Datacubes,  $2^{|\text{Dim}|}$  dimensional combinations have to be examined, each of which originates a cuboid. For each cuboid, the number of output tuples depends on the domain cardinality of the considered combination (Shoshani, 1997), which is denoted by  $|X|$ .

As previously mentioned, all approaches do not deal with original data but instead with coded data (for obvious optimization reasons). Actually each value of a dimensional attribute  $A_i$  is replaced by an integer in the range  $[0 \dots |A_{i-1}|]$  during a preprocessing step (Shoshani, 1997; Beyer & Ramakrishnan, 1999). Under this assumption, the storage space required for preserving a cuboid according to X is:  $4(|\text{Dim}| + 1) |X|$ . The overall space for storing a full Datacube is bounded by:  $2^{|\text{Dim}|} 4(|\text{Dim}| + 1) \text{Max}(|X|), \forall X \in \wp(\text{Dim})$ .

In contrast, PCube generates concise representations of Datacubes. For each cuboid

Table 3. Relation Cube

RowId	DimId	Sum(Quantity)
1	1	27
1	2	22
2	2	5
1	3	2
2	3	25
...		

Table 4. Relation Dimension

DimId	City	Product	Day
1	0	0	0
2	1	0	0
3	0	1	0
4	0	0	1
5	1	1	0
...			

according to a dimensional combination  $X$ ,  $|X|$  tuples are to be computed but each one only requires to store three values (an identifier, the associated aggregated value and the corresponding dimensional combination), each one needing 4 bytes. Thus the storage requirement for a cuboid is:  $12|X|$ .

When compared to the classical representation (used by BUC for example), the latter result is really significant because as soon as the number of dimensions is higher than 2, our concise representation is more compact. Of course this advantage is increased as the set of dimensions is enlarged because Pcube storage requirement, for any cuboid, is independent of the number of dimensions. The latter remark explains results reported in table 5. They give, according to the number of considered dimensions, the percentage of space occupied by our concise representation when compared to a classical Datacube storage. When the number of dimensions is equal to 10, our condensed representation requires 27.2% of the space needed by classical representation (BUC for example) to store a full Datacube, and only 14.2%, for 20 dimensions.

In order to confirm this analytical comparison, it remains to provide experimental results.

## Experimental Evaluations

We choose to compare our approach with the two most concise lossless representations. Through these experiments, our aim is to compare the underlying main memory requirements and the size of the Datacube to be stored. In order to compute the Quotient Cube and Closed Cube, we use the algorithm Close (Pasquier et al., 1999), proved to be very efficient for mining frequent closed patterns, because we have its sources. The computer has a Pentium 4 to 3 GHz with 1 Gb of RAM and runs under Windows XP. Implementations are performed in C++ and compiled with c++ (GCC) 3.3.3 (cygwin).

Table 6 gives the datasets used for experiments. The columns #Attributes and #Tuples stand for the number of attributes and tuples respectively. In the last column, the size in bytes of the dataset is reported (each dimension or attribute is encoded as an integer requiring 4 bytes for any value).

Table 5. %age of storage space for Pcube vs. classical representation

Dim	1	2	3	4	5	6
% of space condensed vs. classical storages	150 %	100%	75%	60%	50%	42.8%
Dim	7	9	10	12	20	25
% of space condensed vs. classical storages	37.5%	33.3%	27.2%	23%	14.2%	11.5%

Table 6. Datasets

Tables	# Attributes	# Tuples of the initial relation	# Tuples of the aggregate relation	Size (KB)
Mushroom	23	8 124	8 124	747.4
Death	5	24 576	389	7.8
TombNecropolis	7	108 665	1 846	51.7
TombObjects	12	78 539	8 278	397.3
Joint_Objects_Tombs	17	95 194	7 643	519.7

Mushroom and Death are datasets widely known in frequent pattern mining (Bayardo, Goethals, & Zaki, 2003). Mushroom provides various characteristics of mushrooms. Death is a dataset gathering information about patient decease with the date and cause. TombNecropolis and TombObjects are issued from archaeological excavation. They encompass a list of necropolises, their tombs and other properties like: the country, the funeral rite, the objects discovered in the tombs and their description. Finally, Joint\_Objects\_Tombs results from the natural join between TombObjects and TombNecropolis according to the identifiers of necropolises and tombs. These private datasets are provided by an archaeological laboratory of Aix-Marseilles University.

**Remark:** The two datasets Mushroom and Joint\_Objects\_Tombs require too much main memory (> 4Go) when computing the Quotient Cube and the Closed Cube with a minimum threshold equal to 1 (all the possible patterns), thus we have to state a minimum threshold equal to 5% and 1%.

Tables 7 and 8 present the results obtained for the algorithms Close and Pcube for the various datasets. The column Max Memory shows in MB the maximal used memory.

Pcube memory requirements are incomparably lower than the ones of Close. Concerning the size of Datacube representations, the best results are obtained for the Closed Cube and then the Quotient Cube. Although more voluminous,

the Partition Cube reduces significantly the size of the Datacube. For instance, the Partition Cube computed for the dataset Mushroom only needs 12% of the space necessary to store the Datacube. In the worst case (few attributes) the gain is about 50%.

The counterpart of storage saving for the Quotient and Closed Cubes is an efficiency deterioration when evaluating OLAP queries. Actually, with these two representations, only a cover of a Datacube is preserved and additional computations are necessary to answer OLAP queries:

- Pcube computes a concise representation of the Datacube which is based on its characterization (DM-Classes, DM-Partitions and their product). In such a representation, a row of the cube (as exemplified in table 3) contains three elements: RowId, DimId and f(M). Moreover, DimId can be encoded as a bit field to avoid the join operation with the relation Dimension (*cf.* table 4) when evaluating OLAP queries. In a similar way, the link with the original relation (through RowId) does not require a join operation but a direct index. When the number of dimensions is less than 32, each attribute value needs 4 bytes and thus each row 12 bytes. The representation includes the original relation. Thus its size is equal to:  $NbRows * 12 + RelationSize$ .
- For the Quotient Cubes and Closed Cubes,

Table 7. Use of memory of Close

Tables	Max Memory (MB)
Mushroom 5%	354.1
Death	8.1
TombNecropolis	12.8
TombObjects	721.0
Joint_Objects_Tombs 1%	36.3

Table 8. Use of memory of Pcube

Tables	Max Memory (MB)
Mushroom	4.8
Mushroom 5%	4.7
Death	2.5
TombNecropolis	2.6
TombObjects	3.7
Joint_Objects_Tombs	4.1
Joint_Objects_Tombs 1%	4.0



the size of any row is obtained by the product of the number of dimensions and the measure in the original relation by 4 bytes (dimensions are encoded as integers). The obtained size is:  $\text{NbRows}' * 4(|\text{Dim}|+1)$ , where  $\text{NbRows}'$  is the number of tuples required for the Quotient Cube or Closed Cube. Let us underline that  $\text{NbRows}' \leq \text{NbRows}$ .

Table 9 illustrates the size of the three studied representations for the various datasets. These results are resumed in figure 4.

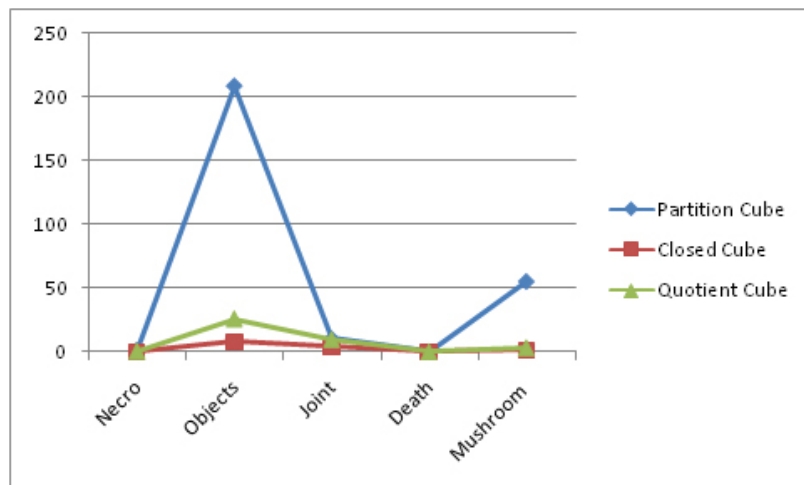
## CONCLUSION

Addressing the issue of Datacube computation and storage is challenging because such a computation needs costly execution time and large main memory space. Datacubes yields huge volume of results, and its storage requires enormous space on disk. In this paper, we focus on the lossless information approaches emphasizing the two ones which propose the most concise representations, namely the Quotient Cube and Closed Cube. We propose an alternative method also providing a storage reduction for the Datacube. Even if the cube reduction is less important than the two previous ones, all the data is stored. Thus, OLAP queries can be answered very efficiently (simple selections in a table) while other approaches require additional

Table 9. Size of the Datacubes

	Size of the Datacube (MB)			
	"Classical"	Partition	Closed	Quotient
TombNecropolis	3.6	1.4	0.1	0.5
TombObjects	903.6	208.9	8	25.8
Joint_Objects_Tombs (1%)	58.8	10.3	4.5	9.7
Death	220 152	117 856	24 984	73 656
Mushroom (5%)	436.8	55.3	1.2	3.3

Figure 4. Size of the Datacubes generated





computations for yielding results. In the worse scenario, when data is very sparse, the size of the Closed and the Quotient Cubes can be as voluminous as the Datacube. On the contrary, when there are more than 2 dimensions, the Partition Cube is always smaller than the data cube. So our approach is a compromise between Datacube storage reduction and efficient execution of OLAP queries. Research perspectives of the presented work are to investigate new issues:

- (1) Taking into account the dimension hierarchies (Hurtado & Mendelzon, 2002) in the very same spirit as Cure for Cubes (Morfonios & Ioannidis, 2006).
- (2) The reduction of the Convex Cubes (Casali, Nejar, Cicchetti, & Lakhal, 2007) and the Emerging Cube (Nedjar, Casali, Cicchetti, & Lakhal, 2007) using partitions.

## REFERENCES

- Bayardo, R., Goethals, B., & Zaki, M. J. (2003). *Workshop on Frequent Itemset Mining Implementations*. Bayardo, Roberto; Goethals, Bart; Zaki, Mohammed J.
- Beyer, K. S., & Ramakrishnan, R. (1999). Bottom-Up Computation of Sparse and Iceberg CUBEs. *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, (pp. 359-370). Dallas, USA.
- Casali, A., Cicchetti, R., & Lakhal, L. (2003a). Cube Lattices: A Framework for Multidimensional Data Mining. *Proceedings of the Third SIAM International Conference on Data Mining (SDM'03)*. San Francisco, USA.
- Casali, A., Cicchetti, R., & Lakhal, L. (2003b). Extracting semantics from data cubes using cube transversals and closures. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, (pp. 69-78). Washington, USA.
- Casali, A., Cicchetti, R., Lakhal, L., & Novelli, N. (2006). Lossless reduction of datacubes. *Proceedings of the 17th international conference on database and expert systems applications (DEXA'06)*, (pp. 409-419). Kraków, Poland.
- Casali, A., Nedjar, S., Cicchetti, R., & Lakhal, L. (2007). Convex cube: Towards a unified structure for multidimensional databases. *Proceedings of the 18th international conference on database and expert systems applications (DEXA'07)*, (pp. 572-581). Regensburg, Germany.
- Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. *Sigmod record*, 26 (1), 65-74.
- Ganter, B., & Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations*. Springer.
- Gilbert, A., Kotidis, Y., Muthukrishnan, S., & Strauss, M. (2001). Surfing Wavelets on Streams : One-Pass Summaries for Approximate Queries. *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, (pp. 79-88). Roma, Italy.
- Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., et al. (1997). Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals. *Data Mining and Knowledge Discovery*, 1 (1), 29-53.
- Gupta, H., & Mumick, I. (2005). Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17 (1), 24-43.
- Han, J., & Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.
- Han, J., Pei, J., Dong, G., & Wang, K. (2001). Efficient Computation of Iceberg Cubes with Complex Measures. *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, (pp. 441-448). Santa Barbara, USA.
- Harinarayan, V., Rajaraman, A., & Ullman, J. (1996). Implementing data cubes efficiently. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD'96)*, (pp. 205-216). Montreal, Canada.
- Hurtado, C. A., & Mendelzon, A. O. (2002). OLAP dimension constraints. *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02)*, (pp. 169-179). Madison, USA.

- Kotidis, Y., & Roussopoulos, N. (1998). An alternative storage organization for rOLAP aggregate views based on cubetrees. *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, (pp. 249-258). Seattle, USA.
- Kotidis, Y., & Roussopoulos, N. (1999). DynaMat: A Dynamic View Management System for Data Warehouses. *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, (pp. 371-382). Philadelphia, USA.
- Kudryavcev, Y. (2006). Efficient algorithms for mOLAP data storage and query processing. In *Spring colloquium for young researchers in databases and information systems, syrcondis*.
- Lakshmanan, L., Pei, J., & Han, J. (2002). Quotient cube: How to summarize the semantics of a data cube. *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, (pp. 778-789). Hong Kong, China.
- Lakshmanan, L., Pei, J., & Zhao, Y. (2003). QC-Trees: An Efficient Summary Structure for Semantic OLAP. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03)*, (pp. 64-75). San Diego, USA.
- Laurent, D., & Spyrtos, N. (1988). Partition semantics for incomplete information in relational databases. *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data (SIGMOD'88)*, (pp. 66-73). Chicago, USA.
- Li, S.-E., & Wang, S. (2005). Semi-closed cube: An effective approach to trading off data cube size and query response time. *Journal of Computer Science and Technology*, 20 (3), 367-372.
- Lopes, S., Petit, J. M., & Lakhal, L. (2002). Functional and Approximate Dependency Mining: Databases and FCA points of View. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 14 (2-3), 93-114.
- Mannila, H., & Toivonen, H. (1996). Multiple Uses of Frequent Sets and Condensed Representations: Extended Abstract. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, (pp. 189-194). Portland, USA.
- Mitchell, T. M. (1996). *Machine learning*. MacGraw-Hill Series in Computer Science.
- Morfonios, K., & Ioannidis, Y. E. (2006). Cure for cubes: Cubing using a rOLAP engine. *Proceedings of the 32nd International Conference on Very Large Data Bases*, (pp. 379-390). Seoul, Korea.
- Nedjar, S., Casali, A., Cicchetti, R., & Lakhal, L. (2007). Emerging cubes for trends analysis in OLAP databases. *Data Warehousing and Knowledge Discovery, 9th International Conference (DaWaK'07)*, (pp. 135-144). Regensburg, Germany.
- Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Discovering Frequent Closed Itemsets for Association Rules. *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*, (pp. 398-416). Jerusalem, Israel.
- Pedersen, T., Jensen, C., & Dyreson, C. (1999). Supporting Imprecision in Multidimensional Databases Using Granularities. *Proceedings of the 11th International Conference on Scientific and Statistical Database Management (SSDBM'99)*, (pp. 90-101). Cleveland, USA.
- Pei, J., Han, J., & Mao, R. (2000). CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. *Proceedings of the 5th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'00)*, (pp. 21-30). Dallas, Texas.
- Ross, K. S. (1997). Fast Computation of Sparse Datacubes. *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB'97)*, (pp. 116-125). Athens, Greece.
- Shanmugasundaram, J., Fayyad, U., & Bradley, P. (1999). Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, (pp. 223-232).
- Shoshani, A. (1997). OLAP and statistical databases: Similarities and differences. *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99)*, (pp. 185-196). Tucson, USA.
- Sismanis, Y., Deligiannakis, A., Roussopoulos, N., & Kotidis, Y. (2002). Dwarf: shrinking the petacube. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD'02)*, (pp. 464-475). Madison, USA.

Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., & Lakhal, L. (2002). Computing Iceberg Concept Lattices with Titanic. *Data and Knowledge Engineering (DKE)*, 42 (2), 189-222.

Theodoratos, D., & Sellis, T. K. (1999). Designing data warehouses. *Data and Knowledge Engineering*, 31 (3), 279-301.

Theodoratos, D., & Xu, W. (2004). Constructing search spaces for materialized view selection. *ACM 7th International Workshop on Data Warehousing and OLAP (DOLAP '04)*, (pp. 112-121). Washington, D.C., USA.

Vitter, J., & Wang, M. (1999). Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD '99)*, (pp. 193- 204). Philadelphia, USA.

Wang, W., Lu, H., Feng, J., & Yu, J. (2002). Condensed Cube: An Effective Approach to Reducing Data Cube Size. *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, (pp. 213-222). San Jose, USA.

Xin, D., Shao, Z., Han, J., & Liu, H. (2006). C-cubing: Efficient computation of closed cubes by aggregation based checking. *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, (p. 4). Atlanta, USA.

Zaki, M. J., & Hsio, C. (2002). CHARM: An Efficient Algorithm for Closed Itemset Mining. *Proceedings of the Second SIAM International Conference on Data Mining (SDM'02)*. Arlington, USA.

## ENDNOTE

- <sup>1</sup> Apart from approaches based on physical techniques.

## APPENDIX: CUBE LATTICE FRAMEWORK

Let  $r$  be a relation over the schema  $R$ . Attributes of  $R$  are divided in two sets (i)  $D$  the set of dimensions, also called categorical or nominal attributes, which correspond to analysis criteria for OLAP, classification or concept learning (Mitchell, 1997) and (ii)  $M$  the set of measures (for OLAP) or class attributes. Moreover, attributes of  $D$  are totally ordered (the underlying order is denoted by  $<_D$ ) and  $\forall A \in D$ ,  $\text{Dim}(A)$  stands for the projection of  $r$  over  $A$ .

The multidimensional space of the categorical database relation  $r$  groups all the valid combinations built up by considering the value sets of attributes in  $D$ , which are enriched with the symbolic value ALL. The latter, introduced in (Gray et al., 1997) when defining the operator Cube-By, is a generalization of all the possible values for any dimension.

The multidimensional space of  $r$  is noted and defined as follows:

$$\text{Space}(r) = \{x_{A \in D}(\text{Dim}(A) \cup \text{ALL})\} \cup \{(\emptyset \dots \emptyset)\}$$

where  $x$  symbolizes the Cartesian product, and  $(\emptyset \dots \emptyset)$  stands for the combination of empty values. Any combination belonging to the multidimensional space is a tuple and represents a multidimensional pattern.

The multidimensional space of  $r$  is structured by the generalization/specialization order between tuples, denoted by  $\preceq$ . This order was originally introduced by T. Mitchell (Mitchell, 1997) in the context of machine learning. In a datawarehouse context, this order has the same

semantic as the operator Rollup/Drilldown (L. Lakshmanan et al., 2002). Let  $u, v$  be two tuples of the multidimensional space of  $r$ :

$$u \preceq v \Leftrightarrow \begin{cases} \forall A \in D \text{ such that } u[A] \neq ALL, u[A] = v[A] \\ \text{or } v = (\emptyset \dots \emptyset) \end{cases}$$

If  $u \preceq v$ , we say that  $u$  is more general than  $v$  in  $\text{Space}(r)$ . In other words,  $u$  captures a similar information to  $v$  but at a rougher granularity level.

**Example 12.** In the multidimensional space of our relation example and by replacing any value by its initial, we have:  $(M, ALL, ALL) \preceq (M, S, d_1)$ , i.e. the tuple  $(M, ALL, ALL)$  is more general than  $(M, S, d_1)$  and  $(M, S, d_1)$  is more specific than  $(M, ALL, ALL)$ . Moreover any tuple generalizes the tuple  $(\emptyset \dots \emptyset)$  and specializes the tuple  $(ALL, ALL, ALL)$ .

The two basic operators provided for tuple construction are: Sum (denoted by  $+$ ) and Product (noted  $\bullet$ ). The Sum of two tuples yields the most specific tuple which generalizes the two operands. Let  $u$  and  $v$  be two tuples in  $\text{Space}(r)$ ,

$$t = u + v \Leftrightarrow \forall A \in D, t[A] = \begin{cases} u[A] \text{ if } u[A] = v[A] \\ ALL \text{ otherwise.} \end{cases}$$

**Example 13.** In our example, we have  $(M, S, d_1) + (M, S, d_2) = (M, S, ALL)$ . This means that the tuple  $(M, S, ALL)$  is built up from the tuples  $(M, S, d_1)$  and  $(M, S, d_2)$ .

The Product of two tuples yields the most general tuple which specializes the two operands. Provided that, for these two tuples, there exists a dimension  $A$  having distinct and real world values (i.e. existing in the original relation), then the only tuple specializing them is the tuple  $(\emptyset \dots \emptyset)$ . Apart from it, the tuple sets which can be used to retrieve them are disjointed. Let  $u$  and  $v$  be two tuples in  $\text{Space}(r)$ , then:

$$t = u \bullet v \Leftrightarrow \begin{cases} t = (\emptyset \dots \emptyset) \text{ if } \exists A \in D \text{ such that } u[A] \neq v[A] \neq ALL, \\ \text{otherwise } \forall A \in D, t[A] = \begin{cases} u[A] \text{ if } v[A] = ALL, \\ v[A] \text{ if } u[A] = ALL. \end{cases} \end{cases}$$

**Example 14.** In our example, we have  $(M, ALL, ALL) \bullet (ALL, S, d_2) = (M, S, d_2)$ . This means that  $(M, ALL, ALL)$  and  $(ALL, S, d_2)$  generalize  $(M, S, d_2)$  and this latter pattern participates to the construction of  $(M, ALL, ALL)$  and  $(ALL, S, d_2)$  (directly or not). The tuples  $(M, ALL, ALL)$  and  $(T, ALL, ALL)$  have as unique common point the tuple of empty values (i.e. the tuple  $(\emptyset \dots \emptyset)$ ).

We define an algebraic structure called cube lattice by endowing the multidimensional space of  $r$  with the generalization order between tuples and using the above-defined operators Sum and Product. Such a structure is a sound start up foundation for solving several multidimensional data mining issues.

**Theorem 3.** Let  $r$  be a categorical database relation over  $D \cup M$ . The ordered set  $CL(r) = \langle Space(r), \preceq \rangle$  is a complete, graded, atomistic and coatomistic lattice, called cube lattice in which Meet ( $\wedge$ ) and Join ( $\vee$ ) elements are given by:

$$\forall t \subseteq CCL(r), \wedge T = +_{t \in T} t,$$

$$\forall t \subseteq CCL(r), \vee T = C(\bullet_{t \in T} t)$$

*Alain Casali obtained the PhD in computer science from the Aix-Marseille Universités (France) in 2005. He is an assistant professor at the University of Aix-Marseille II - IUT of Aix en Provence and is a member of the LIF laboratory. He studies the lattice algorithmic and the multidimensional data mining.*

*Sébastien Nedjar is a PhD student at the LIF laboratory of Marseilles (France). His research work concerns OLAP mining and data warehousing.*

*Rosine Cicchetti is a full professor at the Aix-Marseille Universités (France) and responsible of the database and machine learning research team at the Laboratory of Fundamental Computer Science (LIF) of Marseilles. She obtained the PhD in 1990 (University of Nice-Sophia-Antipolis, France) and the Habilitation for Research Direction in 1996 (University of Aix-Marseilles). Her research topics encompass databases, data mining, data warehousing and statistical databases.*

*Lotfi Lakhal received the PhD in computer science and the Habilitation for Research Direction from the University of Nice-Sophia-Antipolis (France), respectively, in 1986 and in 1991. He is a full professor at the Aix-Marseille Universités- IUT of Aix en Provence and member of the laboratory LIF. His research interest includes databases, formal concept analysis, data mining, data warehousing and data streaming.*

*Noël Novelli received the PhD in computer sciences from the University of Aix-Marseille II (France) in 2000. He was an assistant professor (2001-2004) at the University of Bordeaux (France), and from 2004 he is an assistant professor at the University of Aix-Marseille II – Sciences Faculty – Computer Sciences Department, at the fundamental computer sciences lab of Marseille. His research interest includes databases, data mining, data warehousing and data visualization.*

# A Parameterized Framework for Clustering Streams

*Vasudha Bhatnagar, University of Delhi, India*

*Sharanjit Kaur, University of Delhi, India*

*Laurent Mignet, I.B.M., Indian Research Lab, India*

---

## ABSTRACT

*Clustering of data streams finds important applications in tracking evolution of various phenomena in medical, meteorological, astrophysical, seismic studies. Algorithms designed for this purpose are capable of adapting the discovered clustering model to the changes in data characteristics but are not capable of adapting to the user's requirements themselves. Based on the previous observation, we perform a comparative study of different approaches for existing stream clustering algorithms and present a parameterized architectural framework that exploits nuances of the algorithms. This framework permits the end user to tailor a method to suit his specific application needs. We give a parameterized framework that empowers the end-users of KDD technology to build a clustering model. The framework delivers results as per the user's application requirements. We also present two assembled algorithms G-kMeans and G-dbscan to instantiate the proposed framework and compare the performance with the existing stream clustering algorithms.*

*Keywords:* architecture; clustering; data stream; grid; micro-cluster

---

## INTRODUCTION

Data streams pose special challenges to mining algorithms, not only because of the huge volume of on-line data streams and its computation (Henzinger, Raghavan & Rajagopalan, 1998; Babcock, Babu, Datar, Motwani & Widom, 2002; Carney, Cetintemel, Cherniack, Conway, Lee, Seidman et al., 2002; Domingos and Hulten, 2000), but also because of the fact that data in streams may show temporal correlations. Such temporal correlations help in disclosing

important data trends in XML document clustering (Rusu, Rahayu & Taniar, 2008), multimedia communication and programming support for ubiquitous distributed computing environment (Aggarwal, 2007).

Clustering is considered as one of the most popular and effective techniques for discovering similarity trends in data streams. Compactness of representation, fast incremental processing of new data points, insensitivity to order of input records have been identified as basic requirements in stream clustering algorithms



(Henzinger, Raghavan & Rajagopalan, 1998; Barb'ara, 2002; Orłowska, Sun & Li, 2006).

The problem of incremental clustering is addressed in Zhang, Ramakrishnan & Livny (1996) and inspired clustering of data streams. The importance of the problem is evident from the large body of work (Aggarwal, Han, Wang & Yu, 2003; Motoyoshi, Miura & Shioya, 2004; Park & Lee, 2004) that has evolved over a relatively short period of time since the earliest attempt to address the problem of stream clustering (Guha, Mishra, Motwani & O'Callaghan, 2000).

The algorithms that have been developed for stream clustering have either an on-line or a batch component for processing incoming data, to maintain synopsis. A mechanism is used to highlight the evolving nature of data in stream. Clustering is done using varied approaches based on distance (*k-means* or *k-median*), density estimation, statistical methods (e.g. co-variance, skewness etc.) and connected component analysis.

## Motivation

One of the reasons for the fallen-short-of-anticipated growth curve of KDD technology is that the end-user is forced to use the mining algorithms provided by the data mining packages and has no say in designing the algorithm. The current KDD technology is limited by the adhoc approach for solving individual problems (Yang & Wu, 2006). The need for a unified framework for integrating different data mining tasks has been recognized recently (Yang & Wu, 2006).

Motivated by the above observation, we propose a parameterized framework for stream clustering. The framework empowers the end-user to choose the features of the algorithm to suit their business requirements in terms of nature of inputs, outputs, availability of resources etc.. The proposed component-based architecture of stream clustering algorithms advocates development of a data-mining environment where the user can match the application needs with the features of the components and assemble the

algorithm. The approach overcomes the rigidity prevalent in the use of data mining environments, where the match between the available algorithmic features and desired functionality is sometime less than satisfactory. This work lays the theoretical foundation for the unified framework by parameterizing an algorithm based on application requirements.

## Outline of the Paper

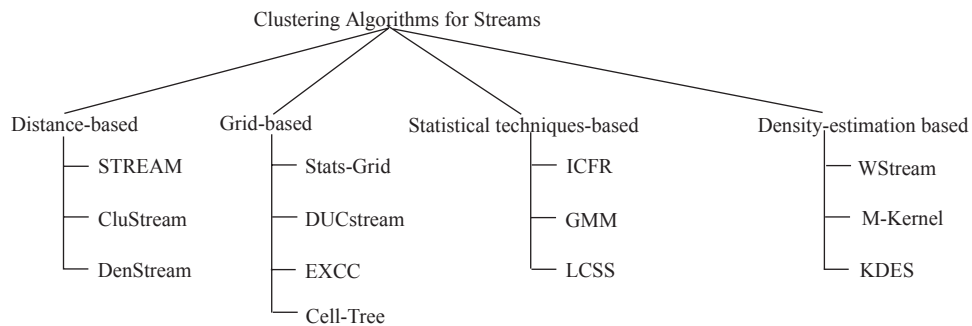
The paper is divided into five sections. Section "Comparison of Stream Clustering Algorithms" studies different approaches used in stream clustering algorithms, and a systematic comparison vis-à-vis the nature of input, output, processing and functionality is presented. The study leads to a component based architectural framework underlying all stream clustering algorithms, which is discussed in Section "Generic Architecture for Stream Clustering Algorithms". Based on this framework, subsection "Architectural Framework" proposes a scheme to assemble designer algorithms by selecting appropriate components to suit the user's specific needs. Section "Realization of the Framework" instantiates the proposed framework by laying down hypothetical user requirements and assembling two algorithms *G-kMeans* and *G-dbscan*. Experimental evaluation of the two algorithms is also presented in the same section.

## COMPARISON OF STREAM CLUSTERING ALGORITHMS

In this section, we describe some of the recent stream clustering algorithms and categorize them based on the underlying approach for clustering (Figure 1).

A closer look reveals that some algorithms qualify for more than one category (e.g. *Density-Stream*), but for the sake of clarity, each algorithm has been placed in the most appropriate category. The salient features of some representative stream clustering algorithms from each category are described in the following section

Figure 1. Categorization of stream clustering algorithms



and a comparison of these features are given at the end of the section. Algorithms for handling noisy and high dimensional data streams have been consciously omitted, since such features are add-ons to the basic problem of clustering data streams.

## Distance-Based Algorithms

In this approach, initially a set of points is selected to represent the center of clusters. Subsequently a distance metric is used to place points in appropriate clusters. Some well-known stream clustering algorithms that use this approach are described below.

- a. *STREAM* algorithm (Guha, Mishra, Motwani & O'Callaghan, 2002), uses the landmark window model that processes stream in batches and generates  $k$  optimal clusters using an approximation approach. Stream is treated as a sequence of chunks (batches), and for each chunk frequency of each distinct point is computed leading to weighted chunks. An approximation algorithm (LOCALSEARCH), which is a *Lagrangian relaxation* of *k-Median* problem is applied on each weighted chunk to retain  $k$  weighted cluster centers. The weight of each cluster center is the sum of the weights of the members in the cluster. Subsequently the same algorithm is applied on retained weighted cluster centers

to get optimal number of clusters for the entire stream. The algorithm is memory efficient and has  $O(nm + nk \log k)$  running time where  $n$  is data size,  $m$  is the number of centers used in computation in a batch, and  $k$  is number of centers retained in each chunk.

- b. *CluStream* (Aggarwal, Han, Wang & Yu, 2003) handles evolving stream using an on-line and an off-line component. The algorithm generates clusters over different portions of stream using pyramidal time frame.

It summarizes information about incoming points in micro-clusters ( $\mu C$ s), which maintain summary information similar to cluster feature vector of Zhang, Ramakrishnan & Livny (1996), except for the additional information about time stamp.

*k-Means* algorithm is applied to a set of first few points in the stream to generate  $k$  micro-clusters ( $\mu C$ s), which form the synopsis. Subsequently, the on-line component absorbs incoming data points into micro-clusters based on distance. If the new incoming point cannot be absorbed in one of the existing  $\mu C$ s, then a new  $\mu C$  is created. It is imperative to ensure that the size of the synopsis remains constant. Thus, either a cluster with few points or least relevance time stamp is deleted, or two clusters with which that are close to each other are merged. The deleted  $\mu C$  is treated as an outlier from the current point of view. Pyramidal time

frame is used to store snapshots of the synopsis at different time instances so that clusters can be discovered in user specified time horizon  $h$  with desired granularity.

The off-line component of the algorithm discovers convex clusters by applying *k-Means* algorithm on all  $\mu C$ s generated in time horizon  $h$ . Subtractive properties of  $\mu C$ s are exploited to generate higher-level clusters from the stored synopsis at different time horizons.

- c. *DenStream* handles the dynamic nature of evolving data streams using a damped window model (Cao, Ester, Qian & Zhou, 2006). This algorithm also has an on-line and an off-line component. The on-line component is used for micro-cluster maintenance and off-line component generates clusters on demand. Potential-microclusters ( $P-\mu C$ s) and Outlier-microclusters ( $O-\mu C$ s) are used for incremental computation by the on-line component for handling dynamics of an evolving data stream. The definitions of  $P-\mu C$  and  $O-\mu C$  are extended definitions of  $\mu C$  used in Aggarwal, Han, Wang & Yu (2003) wherein, summarized information is also weighted with respect to time.

During initialization phase, *dbscan* algorithm (Ester, Kriegel, Sander & Xu, 1996) is applied to initial  $n$  points to generate potential-microclusters ( $P-\mu C$ s). New data points arriving in stream are added to the nearest  $P-\mu C$ , provided, addition does not cause increase in the radius of the  $P-\mu C$  beyond a pre-defined threshold. Otherwise, either a new outlier-microcluster ( $O-\mu C$ ) is created or the incoming point is merged into its nearest existing outlier microcluster ( $O-\mu C$ ). In the latter case, the weight of each  $O-\mu C$  is computed. If the weight is greater than some threshold  $\theta$ , then  $O-\mu C$  is converted into  $P-\mu C$ . Similarly, the weight of each  $P-\mu C$  is checked periodically to ensure that it is still a valid potential microcluster, else it is deleted. The algorithm uses a fading mechanism to reduce impact of older data on current trends.

As the number of  $O-\mu C$ s may continuously increase with time, *pruning* strategy is used to delete real outliers after reporting them to the user. When clustering is demanded, the offline component applies a variant of *dbscan* algorithm on the set of  $P-\mu C$ s. Use of density-based approach for clustering discovers arbitrary shape clusters and makes the algorithm robust.

The major drawback of distance-based approach is that the number of clusters to be generated needs to be defined in advance. Further, clustering results also depend on data ordering. Distance-based algorithms like *CluStream*, *DenStream* do not generate exclusive clusters. Thus, a point that is originally placed in one cluster as per its features may be merged with another cluster due to memory constraint later.

## Grid-Based Algorithms

Grid-based algorithms for stream divide multi-dimensional data space into a set of mutually exclusive equal-size cells to maintain detailed data distribution. While distance-based methods work with numerical attributes, grid-based methods work with attributes of mixed types (Berkhin, 2003). Some of the recent algorithms using this approach are described below:

- a. *Stats-Grid* algorithm (Park & Lee, 2004) generates arbitrary-shape clusters with high processing speed. Incoming data points in stream are inserted in the cells of grid on the basis of distribution statistics. The number of data points in a cell constitutes its support. When the support of a cell becomes greater than a pre-defined threshold, it is partitioned into two cells on the selected dimension. Cell partitioning is done on a dimension having maximum mean or standard deviation, depending on distribution statistics. Cell partitioning helps maintain information about current trends at appropriate granularity levels. To reduce the impact of historical data on current trend, cells are pruned based on support of the

cell and its statistics are added back to its parent cell. Clustering is done on demand using connected component analysis.

- b. *DUCStream* is a grid-based algorithm that treats the stream as a sequence of chunks of the same size and uses connected component analysis for clustering (Gao, Li, Zhang & Tan, 2005). Distribution statistics of incoming data points are maintained in the grid and initial clusters are created using dense cells of the first chunk. At any time, relative density of a cell ( $rd_c$ ) is computed as  $N_c/(m * t)$ , where  $N_c$  is number of points in the cell  $c$ ,  $m$  is size of chunk and  $t$  is the number of chunks accessed so far. A cell is dense if  $rd_c > \delta$  where  $\delta$  is a user-defined threshold. Connected component analysis is used to create initial clusters using dense cells.

For subsequent chunks, newly formed cell are merged with one of the existing cluster if possible. Else a new cluster is generated. Clusters are updated by removing or merging existing clusters to incorporate new cells after each chunk. The clustering result is set of all clusters found in  $t$  chunks of data seen so far. The algorithm misses emerging clusters because it does not consider recent non-dense cells for clustering.

- c. *ExCC* algorithm (Bhatnagar & Kaur, 2007) addresses two important features of clustering viz. exclusiveness and completeness. The on-line component of the algorithm summarizes the incoming data stream in a grid. Each cell in the grid stores the number of points and the average inter-arrival time of data points in the cell. This information is used during grid pruning which needs to be performed either when clustering is demanded or when the grid outgrows the memory.

The second component performs on-demand clustering using connected component analysis and may be run either on-line or off-line, depending on the requirement of the application/user. Prior to clustering, outdated cells are pruned in order to get a current clustering model. This is accom-

plished while ensuring that even a small cluster that showed up in the stream after last clustering, is detected. This feature ensures complete clustering. Exclusive clustering means that at any time, a point belongs to a unique cluster to which it genuinely belongs (Orlowska, Sun & Li, 2006). This algorithm delivers a precise description of the discovered clusters in terms of their boundaries, signatures of seeds etc.

- d. *Cell-Tree* algorithm (Park & Lee, 2007) is an extension of *Stats-Grid* algorithm and aims to overcome the limitation of the latter's scalability. This is achieved by employing two data structures: sibling-list and cell-tree. Initially the multi-dimensional data space is partitioned into fixed number of mutually exclusive equi-sized cells termed as grid cells. The distribution statistics maintained in each cell is diminished by a pre-defined decay rate as time elapses, to reduce the impact of old information on the current clustering scheme.

A sibling-list is used to manage the set of all grid cells in a one dimensional data space. It acts as an index for locating a specific grid cell. After a dense unit cell on one dimensional data space is created, a new sibling list for another dimension is created as a child of the grid cell. This process is recursively repeated for each dimension and it leads to a cell-tree with maximum depth of  $d$ . A unique path in the cell-tree identifies each dense unit grid cell. While clustering, connected component analysis is applied on  $d$ -dimensional dense unit grid cell whose current support is greater than a pre-defined threshold. Although the algorithm is computationally more expensive, it generates arbitrary-shaped clusters with current trends.

Accumulation of data in a grid structure makes grid-based clustering techniques independent of data ordering (Berkhin, 2003). Hence, the clusters generated are not effected by the input order of incoming data points. The main drawback of grid-based clustering tech-

niques is their degraded performance for high dimensional data. In such situations, the number of cells becomes very large and the grid may not fit in the memory. The pruning strategies are used to accommodate the grid in memory. However, frequent and aggressive pruning of grid may result into the loss of patterns.

## Statistical Methods Based Algorithms

Clustering methods based on statistical techniques rely on initial samples to estimate the unknown probabilities and probability densities. These resulting estimates are then used as an approximation of true values for future computations (Duda, Hart & Stork, 2000).

- a. ICFR (Incremental Clustering using F-value by Regression Analysis) proposed in Motoyoshi, Miura & Shioya (2004), claims to give a more accurate clustering for stream, with constraint of treating stream as a sequence of chunks. Each chunk has a constant size ( $t_2 - t_1$ ) over time axis. In the initialization phase, clusters are generated using similarity function applied on initial ( $h - 1$ ) chunks. For each cluster, the center of gravity, variance, regression-coefficient and  $F$ -value is maintained. To reduce the number of clusters, close clusters are combined iteratively if  $F$ -value of newly merged cluster is bigger than the  $F$ -value of each of the candidate cluster. This procedure is repeated till no more clusters can be combined. New incoming points are collected in a chunk referred to as *recent chunk*. Initial clusters in *recent chunk* are discovered and combined with existing clusters. Else clustering is done from scratch using all clusters formed so far using new  $F$ -value. While clustering, only last ( $h - 1$ ) chunks are used to get recent and current clusters.
- b. The algorithm proposed in Song & Wang (2004), detects clusters using *Gaussian Mixture Model (GMM)*. This algorithm incrementally updates the density estimates,

taking into account only the newly arrived data and previously estimated density. This algorithm is based on *Expectation Maximization* technique and uses a cluster merging strategy based on multivariate statistical tests for equality of covariance and mean. Covariance and mean are used as representations for clusters. The benefit of using a covariance matrix is that it is translation invariant and is used for determining the orientation of a cluster.

- c. Algorithm for detecting low complexity clusters by *skewness* and *kurtosis* was proposed by Song & Wang (2006). *Skewness* and *kurtosis* are employed in addition to mean and covariance, to capture underlying distribution. Multivariate skewness is a single non-negative number, which characterizes the asymmetry of a probability distribution ( $PD$ ) and hence represents asymmetry of clusters. Multivariate kurtosis is also a single non-negative number, which is used to measure the peakedness of a  $PD$  and indicates the concentration of a cluster. Clusters are generated using *Expectation Maximization (EM)* algorithm and for each cluster, all required statistics are computed. The algorithm generates low complexity clusters and provides an accurate description of the shape of a cluster.

In order to reduce the number of clusters, merging is done in two phases. In the first phase, two clusters with comparable mean and covariance are merged. Otherwise, skewness and kurtosis of the entire data in both clusters are tested against multivariate normality. If the normality is acceptable, then these two clusters are merged despite inequalities in their mean and covariance. This merging process is repeated till no more clusters can be merged.

Statistical approaches for clustering are efficient if data dimensionality is low and data belongs to single distribution. A major limitation of these approaches is that stream is always processed in batches. Thus net clustering results is influenced by the initial model generated using the initial sample. Since real life applications,



data may belong to different distributions or may evolve with time, these approaches have limited utility.

## Density Estimation Based Algorithms

Given a sequence of independent random variables, identically drawn from a specific distribution, the general *Density Estimation (DE)* problem is to reveal a density function of underlying distribution. This probability distribution is then used to identify dense and sparse regions in data set with local maxima of the probability distribution function taken as cluster centers (Sain, 1994). *Kernel-Density Estimation (KDE)* is a widely studied nonparametric *DE* method and is suited to data mining applications because it does not make any assumption about the underlying distribution. Most of the algorithms in this category use window-based model.

- a. Zhou, Cai, Wei & Qian (2003) propose *M-Kernel* algorithm for estimating probability density function on line with limited memory and in linear time. The basic idea is to group similar data points and estimate the kernel function for the group. This strategy is instrumental in keeping the memory requirement in control because if  $N$  data points have arrived in the stream so far, then number of kernels,  $m$  is very less i.e.  $m \ll N$ . Each *M-Kernel* has three parameters weight, mean and bandwidth. The algorithm works for both landmark and window models, although in the latter case, only an approximation is delivered. The computed kernels are then ranked to identify clusters. The algorithm has been tested for one dimensional data using *Gaussian Kernel*. The major limitation of this approach is that it processes one dimensional data streams and the memory used is very sensitive to the distribution of the data set.
- b. Heinz & Seeger (2006) extend the idea of merging kernels and propose a resource aware algorithm for *KDE* over streaming data. This algorithm uses *Epanechnikow Kernel*, which has a simple form and bounded support, thereby reducing the computational cost (Gray & Moore, 2003). Bandwidth is dynamically computed based on standard deviation in amortized constant time. For each kernel, a counter (for the incorporated points) and min-max value of all points is maintained. Whenever the number of kernels maintained exceeds a threshold, a merging technique, similar to merging technique given in Zhou, Cai, Wei & Qian (2003), is used.
- c. *WStream* algorithm (Tasoulis, Adams & Hand, 2006) extends conventional *KDE* clustering to spatio-temporal data in stream environment, using *Epanechnikow Kernel*. This algorithm maintains a list of windows, each capturing a cluster. The windows are moved, expanded and contracted incrementally depending on the values of the data points that join the clusters. These operations inherently take into account the fading of older data by periodically computing the weight of windows and using it in kernel function. In case a new data point arrives, which does not belong to any of the existing windows (clusters), a new window is created with suitably initialized kernel parameters. Two windows that overlap considerably, are merged. A major drawback of this approach is that with increase in points more windows need to be maintained where each cluster is represented by at least one window. Hence, with increase in the number of clusters more memory is required.

The major drawback of Kernel Density Estimation based approach is that it is computationally very expensive.



## Comparison of Algorithms

In this section we consolidate the strengths and weaknesses of the algorithms mentioned earlier. Table 1 shows a feature wise comparative analysis of algorithms described in previous subsections.

*CluStream* and *DenStream* process incoming data in an on-line manner and are capable of handling evolving data whereas *STREAM* processes incoming data in batches. *STREAM* works on the approximation of entire data stream from the beginning to the end without distinguishing between old and new data. *CluStream* discards outdated data by using relevance time stamp, whereas *DenStream* fades away synopsis on the basis of the arrival rate of data points in it. Convex-shaped clusters are generated by *STREAM* and *CluStream* whereas arbitrary-shaped clusters are generated by *DenStream*.

*Stats-Grid*, *DUCstream* and *ExCC* use grid structure for summarizing incoming points and require less processing time per point as compared with distance-based algorithms. They use connected component analysis for connecting adjacent cells in grid and hence generate arbitrary shaped clusters. *Stats-Grid* and *DUCstream* do not handle evolving stream because they prune cells on the basis of density, whereas *ExCC* prunes cells on the basis of arrival rate of points in each cell. Hence *ExCC* is capable of generating clusters that depict current trends.

*Statistical* and *Density Estimation* based approaches are computationally more expensive than the previous two approaches and cannot handle high speed stream. *Statistical* approaches are suitable for data belonging to single distribution, while *Density Estimation* based approaches are more flexible. Density Estimation based approaches are capable of handling evolving

Table 1. Comparison of Features of some Clustering Algorithms for Streams, DS\*: Distance based, CCA\*: Connected Component Analysis, RA\*: Regression Analysis, DN\*: Density based, C\*: Convex, A\*: Arbitrary, E\*: Elliptical

Features	STREAM	CluStream	Stat-Grid	ICFR	DUC-Stream	Den-Stream	ExCC
Year	2002	2003	2004	2004	2005	2006	2007
Nature of Processing	Batch	Online	Online	Batch	Batch	Online	Online
Pre-defined Number of Clusters	No	Yes	No	No	No	No	No
Initial Phase	Yes	Yes	No	Yes	No	Yes	No
Support for On-demand Clustering	No	Yes	Yes	No	No	Yes	Yes
Evolution Mechanism	No	Yes	No	Yes	No	Yes	Yes
Clustering Technique	DS*	DS*	CCA*	RA*	CCA*	DN*	CCA*
Shape of Cluster	C*	C*	A*	E*	A*	A*	A*
Outlier Detection	No	No	No	No	No	Yes	Yes
Exclusive Clustering	Yes	No	Yes	Yes	Yes	No	Yes

stream. However, both approaches give efficient results for low dimensional data only and the final clustering result is influenced by the initial model generated.

## GENERIC ARCHITECTURE FOR STREAM CLUSTERING ALGORITHMS

This section presents the parameterized framework for stream clustering. Section “Architectural Framework” presents a generic architecture in which the user specifies the requirements of the application, and the framework selects the appropriate components to assemble the algorithm. The later sub-sections describe the tasks in detail and the issues involved in the selection of the components.

### Architectural Framework

Figure 2 shows the architectural framework for assembling a stream clustering algorithm. The framework exploits the task-based generic architecture of stream clustering algorithms that emerges from the comparative study presented in Section “Comparison of Stream Clustering Algorithms”. In the proposed framework, the

end user specifies parameters at two levels. The first set of parameters (user parameters) reflects the specific user needs and is used for assembling the algorithm. This provides flexibility to the user to set goals for each of the tasks. The second set of parameters (algorithmic parameters) is specific to the tailored algorithm.

The component selector selects the components for accomplishing tasks involved in stream clustering. The selection is based on the user parameters. The output of the component selector is a tailored algorithm that accomplishes tasks summarized in Figure 3. Each of the tasks is accomplished using a strategy from some existing algorithm, to meet the user’s requirements in totality.

The user parameters include output needs like shape of the clusters, on-demand/periodic clustering requirement, outlier handling; resources availability in terms of memory and computational power; the nature of the stream in terms of speed, smoothness etc. The ‘Feature’ column of Table 1 is a sample of user requirements that are interpreted as user parameters for component selection. These are the inputs for the selector mechanism to select suitable components from the library. The selected components are subsequently bound to create a tailored stream clustering algorithm.

Figure 2. Architectural framework for assembling stream clustering algorithms

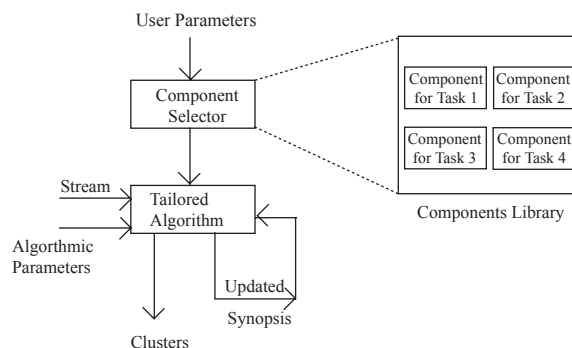


Figure 3. Tasks in a Stream Clustering Algorithm (\* required in some synopsis structures)

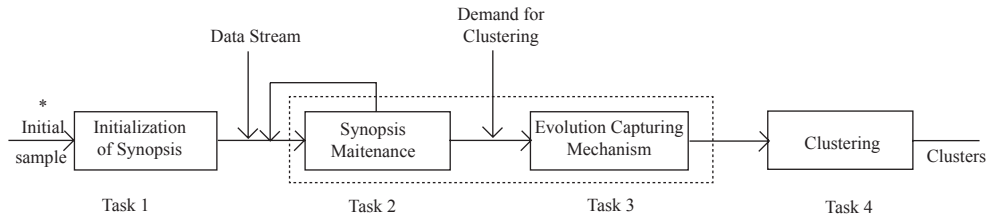


Table 2 gives the possible alternatives for picking up the candidates (parents) for assembling, depending on user level parameters. Once the algorithm has been assembled, the user gives relevant algorithmic parameters, for instance density threshold, number of clusters, intervals, in order to get the desired results.

A wide variety of programming technologies are available for the rapid development of an environment for assembling algorithms in order to implement the framework. Major effort is required in defining the compatible interfaces of the components for seamless integration, leading to a complete algorithm.

## Tasks Involved in Stream Clustering

Figure 3 shows the sequence of the tasks that need to be performed for clustering of data stream. The formulation is based on the comparative study of available stream clustering algorithms. The figure illustrates a generic stream clustering algorithm consisting of four components, each performing one of the tasks mentioned below.

- Selecting the synopsis structure and initializing it (if required)
- Processing the incoming data and updating the synopsis
- Capturing evolution of stream (i.e. detection of evolving and fading clusters)
- Clustering on the synopsis

Initialization of the synopsis structure (task 1) is optional and depends on the clustering algorithm used in task 4. The second component updates synopsis (task 2) while processing incoming points in the stream. To capture changes in data trends, a fading or pruning mechanism is applied on synopsis highlighting the evolution of the new clusters and disappearance of older ones. For some algorithms, the on-line component integrates both these tasks (as indicated by the dotted box). The final component accomplishes the last task using the up-to-date synopsis and delivers the clustering scheme.

We now discuss the issues involved in each of these tasks and their handling in different algorithms. The discussion helps in understanding the issues that arise while interfacing of the components during the assembly.

## Synopsis Structure and its Updation

Since the stream is unbounded and the data is available only for a short duration, it is necessary to maintain a synopsis of the data as it comes. Depending on the clustering algorithm used, the information is summarized in a specific format and is stored as synopsis for future reference. The structure of the synopsis and the clustering algorithm intended to be used, are closely inter-related. In some situations, the synopsis needs to be initialized.

The incoming data points of the stream can be incorporated into the synopsis structure either

Table 2. Sample set of user parameters

Requirement	Type	Candidate Algorithms
Initialization	No	STREAM, Stat-Grid, ICFR, ExCC
	Yes	CluStream, DenStream, DUCStream
Data processing	Batch	STREAM, DUCStream, ICFR
	Online	CluStream, DenStream, Stat-Grid, ExCC
Speed of Stream	Low	STREAM, ICFR
	High	CluStream, DenStream, Stat-Grid, DUCStream, ExCC
Dimensions of data	Low/Moderate	STREAM, CluStream, DenStream, ICFR
	High	Stat-Grid, DUCStream, ExCC
Number of clusters	Known	CluStream
	Unknown	STREAM, DenStream, Stat-Grid, ICFR, DUCStream, ExCC
Shape of Clusters	Convex	STREAM, CluStream
	Arbitrary	DenStream, Stat-Grid, DUCStream, ExCC
	Elliptical	ICFR
Type of clusters	Exclusive	STREAM, Stat-Grid, ICFR, DUCStream, ExCC
	Non-exclusive	CluStream, DenStream
Evolution handling	No	STREAM, Stat-Grid, DUCStream
	Yes	CluStream, DenStream, ICFR, ExCC
Computational requirements	Low	CluStream, DenStream, DUCStream, ExCC
	High	High & STREAM, Stat-Grid, ICFR

in batch mode or in an on-line fashion. The on-line approach requires constant time complexity so that there is no data loss, whereas batch processing does not face this constraint. Batch processing uses the landmark window model to compute clusters over the complete stream received so far. The usage of the landmark window model in clustering makes batch processing unsuitable for capturing evolutionary trends (Aggarwal, Han, Wang & Yu, 2003).

Two commonly used synopsis structures in stream clustering algorithms are i) Set of micro-clusters and ii) Grid.

The algorithms using the set of micro-clusters as synopses or those using Kernel density estimation (Heinz & Seeger, 2006)

need an initialization phase for determining the initial set of clusters. In case of both, the micro-clusters ( $\mu Cs$ ) and fading micro-clusters ( $P-\mu Cs$  and  $O-\mu Cs$ ) (Table 3), new incoming points are absorbed within the closest micro-cluster either in batch or on-line mode. This results into updated synopsis. Time required for maintaining synopsis depends on its size and is an important consideration in the design of the on-line component.

Grid-based synopsis structure maintains detailed data distribution in the data space. The structure does not need initialization and takes constant time for insertion of a new data point (Park & Lee, 2004). This characteristic makes it attractive for handling high speed data

Table 3. Synopsis data structures of some stream clustering algorithms

Features	CluStream	Stats-Grid	ICFR	ExCC	DenStream
Name of Synopsis	Set of Micro-clusters ( $\mu C$ )	Statistical Grid	Set of Initial Clusters	Grid	Set of P- $\mu C$ s and O- $\mu C$ s
Structure of Synopsis	$\overline{CF2^x}, \overline{CF1^x}, CF2', CF1', n$	$(RS_i, C_i, \mu_i, \sigma_i)$	$(\mu, \sigma, fval, RC, DM)$	$n, ts, aat$	$(\overline{CF2^x}, \overline{CF1^x}, wt)$ $(CF2^x, CF1^x, wt, ts)$
Storage Complexity	$O(2d + 3)*N$	$\Omega(K^d)$	$O(N^2)$	$\Omega(K^d)$	$O(2d + 1)*N$
Time Complexity	$O(Nd)$	$\Omega(d)$	$O(N^2)$	$O(d)$	$O(Nd)$
Remarks	$N$ : no. of Microclusters, $n$ : no. of points in each Microcluster $x$ : data value $t$ : time stamp	$RS_i$ : range $C_i$ : count $\mu_i$ : mean $\sigma_i$ : Std. dev. For each dim $i$	$DM$ : distance matrix $RC$ : Regression Coefficient	$ts$ : time stamp $K$ : no. of intervals $aat$ : average arrival time	$N$ : no. of P- $\mu C$ s and O- $\mu C$ s $ts$ : time stamp of O- $\mu C$ s $tw$ : weight

stream. The incoming points are absorbed in the synopsis based on data values along different dimensions. Grid-based synopsis handles high dimensional sparse data more efficiently compared to distance function (Agrawal, Gehrke, Dimitriou & Raghavan, 1998).

### Mechanism to Capture Evolving Data

In order to capture data evolution over time, algorithms employ a recency criterion to highlight current patterns in the stream, and fade out or reduce the impact of older data. Pruning (Aggarwal, Han, Wang & Yu, 2003; Bhatnagar & Kaur, 2007; Park and Lee, 2004) and fading (Cao, Ester, Qian & Zhou, 2006; Motoyoshi, Miura & Shioya, 2004) are two commonly used mechanisms for determining recency.

Pruning can be performed either using time-stamp or data volume, or both. In time-based pruning, the unit of information (e.g. micro-cluster, cell) that has not been updated for a specified period of time is permanently deleted from synopsis structure. The period may either

be explicitly defined as in the sliding window model or may be some pre-defined function of time (Aggarwal, Han, Wang & Yu, 2003). Pruning done on the basis of data volume may miss detection of a slight change in current trend or data distribution (Gao, Li, Zhang & Tan, 2005; Park & Lee, 2004). Data volume-based pruning is thus desirable in applications where only significant distribution changes need to be captured. Some algorithms also use the time of creation of the unit of information in addition to data volume for pruning information, which is not updated for a specified period of time (Bhatnagar & Kaur, 2007).

Fading mechanism dynamically computes the weight of the information in synopsis, based on arrival of conforming data points, as in damped-window model (Cao, Ester, Qian & Zhou, 2006). The importance of historical data is gradually reduced by using a fading function that is typically of the form  $2^{-\lambda \cdot t}$  (Cao, Ester, Qian & Zhou, 2006). This mechanism is preferred when even a slight change in distribution needs to be captured.

## Clustering Technique

Clustering the synopsis is the final task that needs to be performed. Importance of the choice of a clustering technique cannot be undermined since it has a direct impact on the nature of the output (e.g. shape of the cluster), resource requirement (size and structure of the synopsis) and the efficiency of the overall algorithm (design of the on-line/batch component). There is a wide variety of clustering techniques available in literature (Hartigan, 1975; Jain, Murty & Flynn, 1999). Many of these techniques have been suitably modified for clustering data streams.

*k-Median*, *k-Means* and neighborhood density search are commonly used approaches for clustering streams. The *k-Median* based *LOCASEARCH* algorithm is used in *STREAM* (Guha, Mishra, Motwani & O'Callaghan, 2002) at two stages. It initially maintains weighted cluster centers for each chunk and subsequently generates optimal number of clusters from these weighted centers.

*CluStream* uses *weighted k-Means* algorithm for discovering (macro-)clusters when demanded by the user. *DenStream* applies a variant of *dbscan* (Ester, Kriegel, Sander & Xu, 1996) on the synopsis to get arbitrary shaped clusters. Unlike *CluStream*, it does not require pre-defined number of clusters to be generated

and reports all density-connected and density-reachable core- $\mu$ Cs as clusters.

In the grid based approach for clustering, connected component analysis is performed on selected cells to deliver arbitrary shaped clusters. Cells to be clustered are selected based on pruning/fading criteria, as applicable. *Stats-Grid* algorithm selects cells based on points whereas *ExCC* algorithm uses points as well as time for selection. *DUCStream* does clustering on dense cells formed in the first chunk. This algorithm uses an incremental approach and keeps on updating existing clusters by merging all new dense cells formed in subsequent chunks. A new cluster is formed only when a cell cannot be merged within existing clusters.

Statistical approaches use various statistical measures like mean, median, variance, co-variance, regression co-efficient etc. to generate a mathematical model for clustering. This mathematical model is subsequently updated to incorporate new trends. Because of extensive computation, this approach always clusters offline and gives efficient results for a low dimensional data set. *ICFR* uses regression analysis for capturing clusters with local trends. It delivers non-convex shaped and exclusive clusters.

Table 4 summarizes the clustering techniques used in the chosen set of algorithms, along with their respective time complexities.

Table 4. Comparison of clustering components of selected algorithms

Algorithm	STREAM	CluStream	DenStream	Stat-Grid & ExCC	ICFR
Technique Used	<i>k-Median</i>	<i>k-Means</i>	<i>dbscan</i>	CCA	RA
Complexity	$O(NM+Nk\log k)$	$O(Nkt)$	$O(N^2)$	$O(N!)$	$O(N^2)$
Remarks	<i>N</i> : Data size <i>M</i> : No. of Outliers <i>k</i> : No. of Centers	<i>N</i> : No. of $\mu$ Cs <i>t</i> : No. of iterations <i>k</i> : No. of centers	<i>N</i> : No. of core- $\mu$ Cs	<i>N</i> : No. of cells	<i>N</i> : No. of initial clusters



## REALIZATION OF THE FRAMEWORK

To demonstrate the application of the framework, we assemble two stream clustering algorithms as per two different sets of user requirements. The rationale for selecting the components is outlined and experimental evaluation is reported. Experiments reveal that both the assembled algorithms perform comparably with the parent algorithms, from which components have been drawn.

### G-kMeans Algorithm: Example 1

Table 5 lists the requirements of a user. Table 2 helps in determining that *CluStream* satisfies requirements (1), (2), (5), (6) and (9) while *ExCC* satisfies requirements (2), (3), (4), (7), (8) and (9).

Since *CluStream* is a distance based algorithm with on-line component of complexity  $O(Nd)$  (Table 3), it may not be able to handle high speed, high dimensional data stream. Further, *CluStream* also does not lead to exclusive and complete clustering. Both these requirements are met by *ExCC* algorithm, which also delivers arbitrary shaped large number of clusters. To overcome this situation, we assemble an algorithm *G-kMeans* that uses grid structure

for synopsis and *k-means* algorithm for clustering. The choice is made due to the following reasons :

- Per point processing time in grid is lesser than that in micro-cluster approach.
- A complete and desired number of exclusive clusters need to be generated.

When user demands clustering in *G-kMeans*, the grid is pruned to remove the effect of old data on current trends. Each cell in grid is then used by the clustering component, which uses *k-Means* as in *CluStream*.

### G-dbscan Algorithm: Example 2

Considering the user requirements given in Table 6 with reference to Table 2, it was found that *DenStream* satisfies all requirements except (3), (4) and (7).

*DenStream* is also a distance based algorithm with the on-line component of complexity  $O(Nd)$  (Table 3) and may not be able to handle high speed, high dimensional data stream. *DenStream* generates non-exclusive clusters on demand because potential micro-clusters are merged to compute real clusters.

As discussed in previous section, these requirements are met by *ExCC*. We assemble

Table 5. First set of user' requirement for clustering streams

1.	Initialization	Yes
2.	Clustering Requirements	On Demand
3.	Speed of Stream	High
4.	Dimensionality of Data	High
5.	Number of Clusters	Known
6.	Shape of Clusters	Convex
7.	Type of Clusters	Exclusive
8.	Type of Clustering	Complete
9.	Evolution Handling	Yes

Table 6. Second set of user' requirement for clustering streams

1.	Initialization	Yes
2.	Clustering Requirements	On Demand
3.	Speed of Stream	High
4.	Dimensionality of Data	High
5.	Number of Clusters	Unknown
6.	Shape of Clusters	Arbitrary
7.	Type of Clusters	Exclusive
8.	Evolution Handling	Yes
9.	Type of Clustering	Complete
10.	Reporting Outliers	Yes

an algorithm *G-dbscan* that uses grid structure as synopsis and *dbscan* algorithm for clustering as used in *DenStream*.

*G-dbscan* uses grid for storing summarized information about incoming data points. The grid is pruned just before clustering as done in *ExCC*. Each cell in the grid is used as a pseudo point, represented by its signature, while clustering using *dbscan*.

## Experimental Analysis of *G-kMeans* and *G-dbscan*

In this section, we describe the experiments to demonstrate two important features of the assembled algorithms. We show that the quality delivered by the assembled algorithm and time taken are comparable to those of the clustering schemes delivered by the parent algorithms. If both scalability and quality requirements are comparable, meeting the requirements of the user by the assembled algorithm, is certainly an added advantage. The results give us the confidence that the components used for assembling the algorithm do not lead to any deterioration.

All experiments are performed on the Intel Centrino processor with 256 MB RAM, running stand-alone Linux (kernel 2.4.22-1). The algorithm is implemented in ANSI C with no optimizations, and compiled using a g++ compiler (3.3.2-2). In the experiments with streaming data, the results are averaged over multiple runs with total number of data points remaining the same. The following data sets are used:

- a. Network Intrusion Detection Data used in kdd cup, 1999, is available at UCI KDD archive. The data consists of 494,021 records, each having 42 attributes (34 continuous and 8 categorical). Each record corresponds to either normal class or an attack class. The experiments were performed with 23 classes using 34 continuous attributes.
- b. Forest Cover type Data, acquired from UCI KDD archive, has been widely used in clustering experiments. This data has

581,012 observations, each with 54 attributes. There are seven classes in this data set and all ten quantitative attributes are used in the reported experiments.

- c. Synthetic Data generated by *Enclus* data generator (Cheng, Fu & Zhang, 1999) has been used in some experiments. The generated data sets are denoted as *ENdDc-CrR* indicating that there are *d* dimensions (*D*), *c* clusters (*C*) and *r* \* 1000 records (*R*) and are used for verification of proposed assembly approach.

## Evaluation of Cluster Quality of *G-kMeans*

Two measures viz. cluster purity and squared sum of distance (SSQ), were used to evaluate the quality of clusters delivered by *G-kMeans*. The results were compared with those of *CluStream* and *ExCC*. Synthetic data (EN8D10C100R) generated by *Enclus* and Network intrusion data was used for these experiments.

All the three algorithms gave cluster purity above 99.8% on synthetic data establishing the feasibility and comparable quality of stream clustering algorithm tailored using the proposed framework. Average cluster purity of *G-kMeans* with *CluStream* and *ExCC* was compared by recreating the experiment reported in Aggarwal, Han, Wang & Yu (2004), and using the results reported therein. Figure 4 shows that cluster purity of *G-kMeans* is better than *CluStream* but lower than *ExCC*. Marginal lowering of cluster purity of *G-kMeans* can be attributed to the use of distance function for clustering, which is generally not recommended for high-dimensional data set.

The cluster quality of *k-Means* and *G-kMeans* was compared with respect to SSQ, as shown in Figure 5. All records were used in the experiment without treating them as stream in this experiment. The high quality of *G-kMeans* is attributed to the detailed data distribution information summarized in grid. Table 7 shows the comparison of SSQ of the clustering schemes delivered by *G-kMeans* and *CluStream*, using the results reported in Aggarwal, Han, Wang

& Yu (2003). We get mixed results, though we appreciate better compactness achieved by *CluStream* because of the micro-cluster based approach.

### Testing Scalability of *G-kMeans*

The scalability of a stream clustering algorithm needs to be examined with respect to both time and memory requirement. Time complexity is crucial for the on-line component to avoid loss of data in a high speed stream. High scalability is desirable for the offline component so that clustering results are delivered in reasonable time, when demanded by the user. Scalability of the on-line component of *G-kMeans* is not an issue since incoming points are processed in constant time. Therefore, the scalability of the offline component is tested for different granularity levels (Interval) of the grid.

Time required by the component is influenced by the number of cells in the grid, which are used as data points in *G-kMeans*. Figures 6, 8, 10 show the increasing number of grid cells with data points, while Figures 7, 9, 11 show the corresponding clustering times. Though the theoretical time complexity of *k-Means* is  $O(nkt)$ , the actual times are influenced by the data distribution. As seen in Figure 7, the time taken by the *G-kMeans* at finer granularity (In-

terval=14) decreases at 200,000 data points. This dip is explained by change in data distribution leading to pruning of large number of cells and consequent reduction in the iterations required by *k-Means* to generate cluster.

In case of Forest Cover data, though the number of cells in the grid show a constant increase for different grid granularities (Figure 8), the clustering timings do not show a regular pattern (Figure 9). This can be explained by changing data distribution leading to unpredictable pruning and consequent reduction in clustering time. Figures 10, 11 show the observation on Network intrusion data.

Scalability of clustering component of *G-kMeans* algorithm was tested by varying number of input clusters in synthetic data. Figure 12 shows that the clustering time is linear with respect to increase in the number of clusters.

### Evaluation of Cluster Quality in *G-dbscan*

Cluster purity was used as the measure to evaluate the quality of clusters generated by *G-dbscan*. Synthetic data was used to compare *G-dbscan* with *dbscan*. Both algorithms gave approximately 99.8% purity on this data.

The experiment reported in Cao, Ester, Qian & Zhou (2006) was recreated to compare

Figure 4. Cluster purity comparison (Network Intrusion Data)

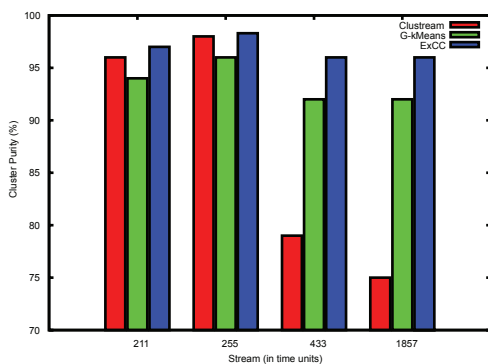


Figure 5. Comparison of cluster quality (SSQ) of EN8D10C100R, Interval = 14

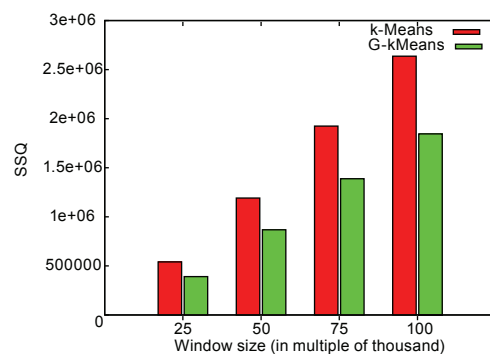


Table 7. Comparison of cluster quality (SSQ) of network intrusion data

Points (in thousands)	CluStream	G-kMeans
150	1E+13	1E+12
250	1E+5	1E+8
350	1E+12	1E+9
450	1E+8	1E+8

Figure 6. Increase in grid cells with number of points (EN5D10C200R)

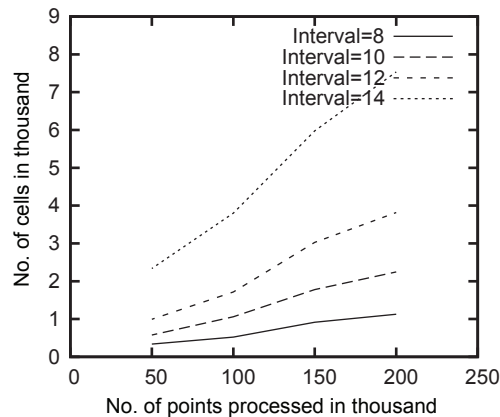


Figure 7. Clustering time (EN5D10C200R)

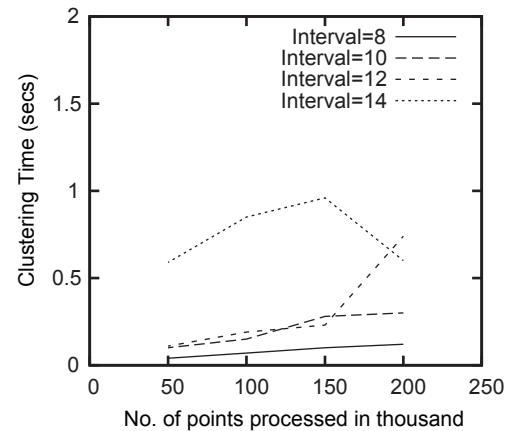


Figure 8. Increase in grid cells with number of points (forest cover data)

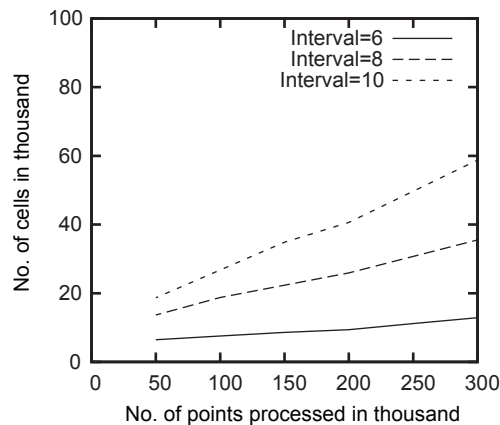


Figure 9. Variation in clustering time (forest cover data)

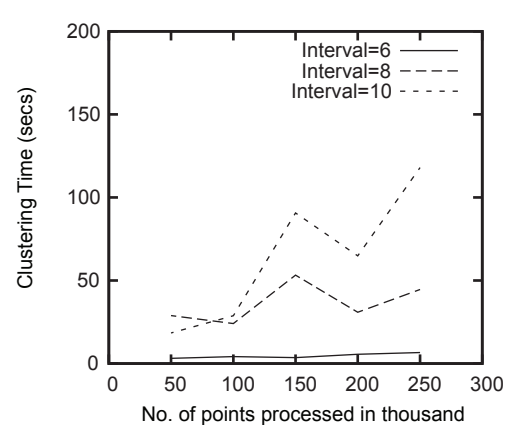


Figure 10. Increase in grid cells with number of points (network intrusion data)

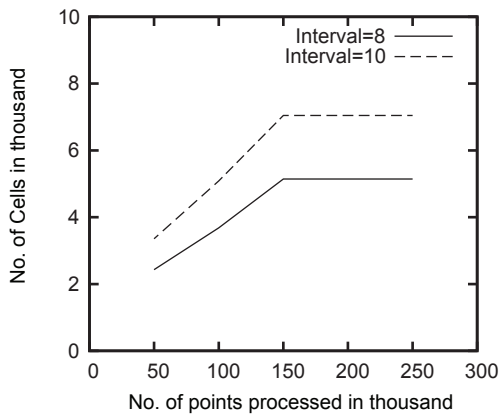


Figure 11. Variation in clustering time (network intrusion data)

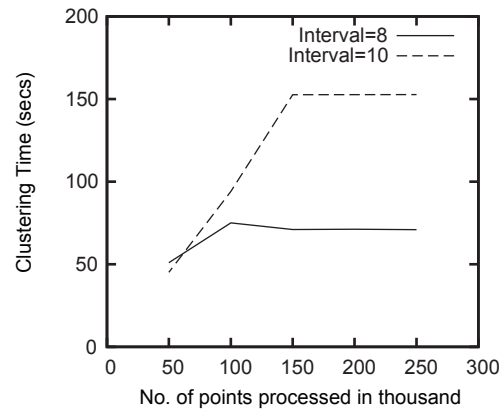
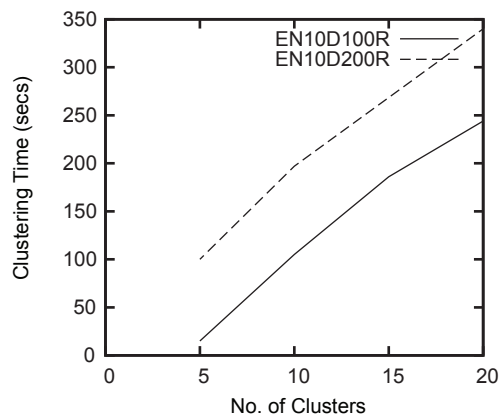


Figure 12. Scalability of cluster component using synthetic data (Interval = 8)



cluster quality of *DenStream* and *G-dbscan* on Network Intrusion data. Figure 13 shows that *G-dbscan* outperforms *DenStream* at every time unit.

### Testing Scalability of *G-dbscan*

The results of the experiment on Network Intrusion data for comparing the scalability of *CluStream*, *DenStream* and *G-dbscan* are shown in Figure 14. The execution times of *CluStream* and *DenStream* grow linearly as stream proceeds, whereas *G-dbscan* executes stream in constant time because of grid structure.

## CONCLUSION

This paper presents a parameterized framework for assembling a stream clustering algorithm. The framework is based on the generic architecture underlying the existing algorithms and is motivated by the need to overcome the adhoc approach for designing algorithms to solve individual problems. The proposed framework enhances the user involvement in the use of the KDD technology, by accepting the business requirements as high level parameters, and tailoring the algorithm best suited for the application needs.

The two algorithms *G-kMeans* and *G-dbscan* are presented to instantiate the proposed framework based on two different user requirements sets. Experimental evaluation of the two algorithms with respect to the quality of clustering and scalability indicate viability of the proposal.

## REFERENCES

- Aggarwal, C.C. (2007). 'Data Streams: Models and Algorithms', Springer Science+Business Media, New York.
- Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). 'A Framework for Clustering Evolving

Figure 13. Cluster purity comparison

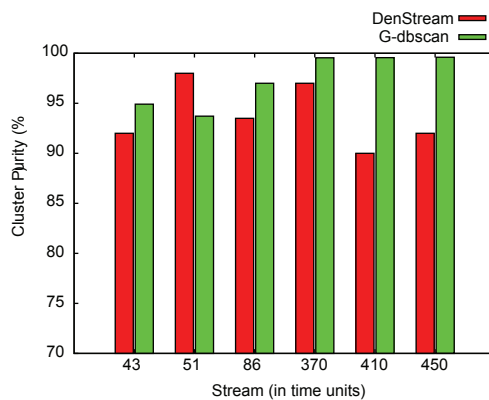
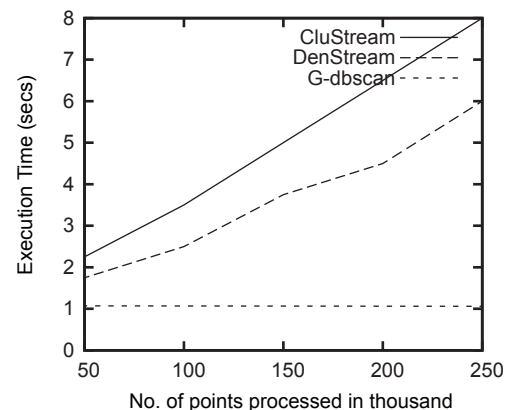


Figure 14. Execution time vs. stream length (network intrusion detection data, interval = 12)



Data Streams', In Proceedings of the 29th VLDB conference, pp.81–92.

Aggarwal, C. C., Han, J., & Yu, P. S. (2004). 'A Framework for Projected Clustering of High Dimensional Data Streams', In Proceedings of International Conference on Very Large Data Bases, Toronto, Canada, pp.852–863.

Agrawal, R., Gehrke, J., Dimitriou, G., & Raghavan, P. (1998). 'Automatic Subspace Clustering of High Dimensional data for Data Mining application', ACM SIGMOD Record, pp.94–105.

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). 'Models and Issues in Data Stream Systems', In Proceedings of the ACM Symposium on Principles of Database Systems (PODS), pp.1–16.

Barb'ara, D. (2002). 'Requirements of Clustering Data Streams', ACM SIGKDD Explorations Newsletter, Vol. 3, pp.23–27.

Berkhin, P. (2003). 'Survey of Clustering Data Mining Techniques', Accure Software Inc.

Bhatnagar, V., & Kaur, S. (2007). 'Exclusive and Complete Clustering of Streams', In Proceedings of the International Conference on Database and Expert Systems Applications, Germany, pp.629–638.

Cao, F., Ester, M., Qian, W., & Zhou, A. (2006). 'Density-Based Clustering over an Evolving Data Stream with Noise', In Proceedings of the SIAM Conference on Data Mining, pp.326–337.

Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G. et al. (2002). 'Monitoring Streams: A New Class of Data Management Applications', In Proceedings of the 28<sup>th</sup> International Conference on Very Large Data Bases, Hong Kong, China, pp.215–226.

Cheng, C. H., Fu, A. W., & Zhang, Y. (1999). 'Entropy Based Subspace Clustering for Mining Numerical Data', In Proceedings of 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.84–93.

Domingos, P., & Hulten, G. (2000). 'Mining High-Speed Data Streams', In Proceedings of 6<sup>th</sup> International Conference on Knowledge Discovery and Data Mining, pp.71–80.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). 'Pattern Classification', John Wiley and Sons.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). 'A Density-based Algorithm for Discovering Clusters in Large Spatial DBs with Noise', In Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining.

Gao, J., Li, J., Zhang, Z., & Tan, P. (2005). 'An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection', In Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hanoi, pp.420–425.

Gray, A., & Moore, A. W. (2003). 'Nonparametric Density Estimation : Towards Computational Trac-



- tability', In Proceedings of SIAM International Conference on Data Mining.
- Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2000). 'Clustering Data Streams', In IEEE Symposium on Foundations of Computer Science, pp.259–366.
- Guha, S., Mishra, N., Motwani, R., & O'Callaghan, L. (2002). 'Streaming-Data Algorithms for High-Quality Clustering', In Proceedings of IEEE International Conference on Data Engineering.
- Hartigan, J. A. (1975). 'Clustering Algorithms', John Wiley & Sons Inc.
- Heinz, C., & Seeger, B. (2006). 'Towards Kernel Density Estimation Over Streaming Data', In Proceedings of 13th International Conference on Management of Data (COMAD), Delhi, India.
- Henzinger, M. R., Raghavan, P., & Rajagopalan, S. (1998). 'Computing on Data Streams', Technical Note 1998-011, Digital Systems Research Center, Palo Alto, California.
- Jain, A. K., Murty, M. N., & Flynn, P. L. (1999). 'Data Clustering : A Review', ACM Computing Survey, pp.264–323.
- Motoyoshi, M., Miura, T., & Shioya, I. (2004). 'Clustering Stream Data by Regression Analysis', In Proceedings of the 2<sup>nd</sup> workshop on Australasian Information security, Data Mining and Web Intelligence, and Software Internationalization, Dunedin, New Zealand, pp.115–120.
- Orlowska, M. E., Sun, X., & Li, X. (2006). 'Can Exclusive Clustering on Streaming Data be Achieved', ACM SIGKDD Explorations Newsletter, Vol 8, pp.102–108.
- Park, N. H., & Lee, W. S. (2004). 'Statistical Grid-based Clustering over Data streams', ACM SIGMOD Record, Volume 33, pp.32–37.
- Park, N. H., & Lee, W. S. (2007). 'Cell trees: An Adaptive Synopsis Structure for Clustering Multi-dimensional On-line Data Streams', Data and Knowledge Engineering, Vol 63, pp.528–549.
- Rusu, L. I., Rahayu, W., & Taniar, D. (2008). 'Intelligent Dynamic XML Documents Clustering', In Proceedings of the 22<sup>nd</sup> International Conference on Advanced Information Networking and Applications, Japan, pp 449-456.
- Sain, S. R. (1994). 'Adaptive Kernel Density Estimation', PhD Thesis, Rice University.
- Song, M., & Wang, H. (2004). 'Incremental Estimation of Gaussian Mixture Models for Online Data Stream Clustering', In International Conference on Bioinformatics and its Applications, Hong Kong.
- Song, M., & Wang, H. (2006). 'Detecting Low Complexity Clusters by Skewness and Kurtosis in Data Stream Clustering', In 9th International Symposium on AI and Maths, Florida.
- Tasoulis, D. K., Adams, N. M., & Hand, D. J. (2006). 'Unsupervised Clustering in Streaming Data', In Proceedings of 6th IEEE International Conference on Data Mining -Workshops, pp.638–642.
- UCI KDD Archive, <http://kdd.ics.uci.edu>.
- Yang, Q., & Wu, X. (2006). '10 Challenging Problems in Data Mining Research', International Journal of Information Technology & Decision, Vol 5, pp.597–604.
- Zhang, T., Ramakrishnan, R., & Livny, M. (1996). 'BIRCH: An Efficient Data Clustering Method for Very Large Databases', In Proceedings of the ACM SIGMOD International Conference on Management of Data, Quebec, Canada, pp.103–114.
- Zhou, A., Cai, Z., Wei, L., & Qian, W. (2003). 'M-Kernel Merging : Towards Density Estimation Over Data Streams', In Proceedings of 8th International Conference on Database Systems for Advanced Applications (DASFAA), pp.285–292.

*Vasudha Bhatnagar did her master's in computer applications from University of Delhi, Delhi, India in 1985. She completed her doctoral studies from Jamia Millia Islamia, New Delhi, India in 2001. She worked in C-Dot from 1985 - 1989 as a software engineer. Thereafter she taught in Moti Lal Nehru College, University of Delhi (1989 - 2002). She is currently a reader in Department of Computer Science, University of Delhi, Delhi, India. Her broad area of interest is intelligent data analysis. She is particularly interested in developing process models for knowledge discovery in databases and data mining algorithms, problems pertaining to modeling of changes in discovered knowledge in evolving (streaming) data sets, handling user subjectivity in KDD, projected clustering, outlier detection.*

*Sharanjit Kaur received her master's degree in computer applications in 1994 and bachelor's degree in computer science in 1991. She started teaching in 1995. Currently she is pursuing her doctoral studies in the area of 'Data Mining' from Department of Computer Science, University of Delhi, Delhi, India. Her research interest spans the area of Data Mining, Databases and Operating System.*

*Laurent Mignet received his MS in computer science (1997) from Paris VI/CNAM/ENST and PhD in Database (2001) from INRIA/CNAM. He then spent two years at the Department of Computer Science of University of Toronto with an INRIA fellowship the first year. Since February 2004 he joined IBM research in Delhi. His research works resulted to international publications in journals and top conferences as well as patents filled in USPTO. Some of his work has been transferred to the industry in a 2 startup during his PhD and now in IBM products. He is DB2 subject matter expert as well as IBM advocate for the information management brand for some IBM customers.*

# A Hybrid Method for High-Utility Itemsets Mining in Large High-Dimensional Data

Guangzhu Yu, Donghua University, China

Shihuang Shao, Donghua University, China

Bin Luo, Guangdong University of Technology, China

Xianhui Zeng, Donghua University, China

---

## ABSTRACT

*Existing algorithms for high-utility itemsets mining are column enumeration based, adopting an Apriori-like candidate set generation-and-test approach, and thus are inadequate in datasets with high dimensions or long patterns. To solve the problem, this paper proposed a hybrid model and a row enumeration-based algorithm, i.e., Inter-transaction, to discover high-utility itemsets from two directions: an existing algorithm can be used to seek short high-utility itemsets from the bottom, while Inter-transaction can be used to seek long high-utility itemsets from the top. Inter-transaction makes full use of the characteristic that there are few common items between or among long transactions. By intersecting relevant transactions, the new algorithm can identify long high-utility itemsets, without extending short itemsets step by step. In addition, we also developed new pruning strategies and an optimization technique to improve the performance of Inter-transaction.*

**Keywords:** high-utility itemset; hybrid model; large high-dimensional data; partition method

---

## INTRODUCTION

Traditional association rule mining (ARM) assumes that the important must be frequent and aims at discovering frequent itemsets. However, in the real world, the frequent is not necessarily important; some infrequent itemsets may have high utility values and thus are important to users. For example, in a transaction database, there

are 1000 sale records of milk which occupy 10% of the total transaction number, contributing 1% of the total profit. In the meantime, there are 600 sale records of birthday cake that occupy 6% of the total transaction number, contributing 5% of the total profit. If the support threshold is 8%, according to traditional algorithms for frequent itemset mining, milk will be reported as a frequent itemset and birthday cake will be

ignored. But in fact, market professionals must be more interested in birthday cake because it contributes a larger portion to total profit than milk. The example shows that support is not sufficient to reflect users' interests and such mining results might not be satisfactory.

According to Expectancy Theory (Vroom, 1964), we have the well-known equation "motivation = probability \* utility", which says that motivation is determined by the utility of making a decision and the probability of success. In retailing field, users are not only interested in the frequency of occurrence of an itemset (support), but also its utility. So a decision-oriented ARM algorithm should output both the support and the utility of interesting patterns. For this reason, utility-based ARM (or utility mining for short) has been proposed to discover all the itemsets in a database with utility values higher than a user-specified threshold.

Utility of an item is a subjective term, depending on users and applications; it could be measured in terms of profit, cost, risk, aesthetic value or other expressions of user preference. For easy understanding, in this paper, "utility" is viewed as economic utility such as sales profit, and all databases are regarded as transaction databases, so that we can define the utility of an item as the product of quantity sold and the unit profit of the item. Table 1 is an example of a simplified transaction database where the total utility value is 162. The number in each transaction in Table 1 is the sales profit of each item. If  $s(X)$  and  $u(X)$  represent the support and utility of itemset  $X$  respectively (for details, refer to definition 4 in Section 2), then  $u(A,B)=43$ ,  $s(A,B)=5$ ,  $u(A,B,C)=54$ ,  $s(A,B,C)=3$ ,  $u(A,B,C,D)=45$ ,  $s(A,B,C,D)=2$ ,  $u(A,B,C,D,E)=57$ , and  $s(A,B,C,D,E)=2$ .

If the support threshold is 3 and the utility threshold is 50,  $\{A, B\}$  is a frequent but not a high-utility itemset. On the other hand,  $\{A,B,C\}$  is both a frequent and high-utility itemset,  $\{A,B,C,D\}$  is neither a frequent nor a high-utility itemset and  $\{A,B,C,D,E\}$  is a high-utility but non-frequent itemset.

From the above example, we can draw a conclusion: downward closure property, which

states that if an itemset is frequent by support, all its nonempty subsets must also be frequent by support, does not apply to utility mining. Relevant studies have shown that utility constraint is neither anti-monotone, monotone, succinct, nor convertible (Shen, Zhang, & Yang, 2000; Yao, Hamilton, & Geng, 2006). Because of this property, most algorithms for frequent pattern mining such as FP-Tree (Han, Pei, Yin, & Mao, 2004), CARPENTER (Pang, Cong, Tung, Yang, & Zaki, 2003), **Tree-projection** (Agarwal, Aggarwal, & Prasad, 2000) and so on can not be used to find high-utility itemsets.

Lots of researches have been conducted to improve the usefulness of traditional ARM. Ngan, Lam, Wong, and Fu (2005) proposed an algorithm called COFI+BOMO to mine N-most interesting itemsets, but the interestingness measure still depends on the support. Value added association rule (Wang, Zhou, & Han, 2002; Lin, Yao, & Louie, 2002) extends traditional association rule by taking the semantics of data into consideration. The difference between Wang et al. (2002) and Lin et al. (2002) is that price and quantity of supermarket sales are considered in the former, while the latter tries to attach a value to every item in the database and use the added values to rank the association rule. Weighted association rule gives up treating all the items and all the transactions uniformly

Table 1. A transaction database

	A	B	C	D	E
T1	0	0	5	0	1
T2	2	3	0	0	0
T3	3	5	15	7	4
T4	0	0	4	7	2
T5	4	5	8	0	0
T6	9	4	0	0	2
T7	6	0	8	3	6
T8	0	0	0	6	3
T9	3	0	0	9	5
T10	3	5	6	1	8

by assigning different weights to items (Cai, Fu, Cheng, & Kong, 1998) or transactions (Lu, Hu, & Li, 2001). These weights essentially reflect the users' preferences. Quantitative association rule mining (Aumann & Lindell, 2003; Webb, 2001) introduces statistical inference theory into the data mining field to find extraordinary phenomena in the database. In order to provide efficient data initialization for mining association rules in data warehouse, Tjioe and Taniar (2005) proposed four algorithms, which focus on quantitative attributes.

To find frequent high-utility itemsets, Shen et al. (2002) proposed an objective-oriented Apriori (OOApriori) model by putting utility constraint into the Apriori algorithm. Chan, Yang, and Shen (2003) proposed an algorithm to mine top-k frequent high-utility closed patterns. To reduce search space, Chan et al. (2003) developed a new pruning strategy based on a weaker but anti-monotonic condition to prune low-utility itemsets. Barber and Hamilton (2003) use itemset share as a measure to overcome the lack of support. Item share is defined as a fraction of some numerical values. It can reflect the impact of the sales quantities of items on the cost or profit of an itemset, and thus it should be regarded as a utility.

All the algorithms mentioned above are utility-related, but none of them are utility-based. To the best of our knowledge, only UMining (Yao & Hamilton, 2006) and Two-phase (Liu, Liao, & Choudhary, 2005) can be used to find high-utility itemsets. In UMining algorithm, the utility upper bound property is used to reduce the size of the candidate set. In addition, the support upper bound property is used in a heuristics model to predict whether an itemset should be added into the candidate set. Unfortunately, the heuristics model can not guarantee an accurate prediction. In Two-phase algorithm, transaction weighted downward closure property is used to reduce search space. The algorithm often overestimates the utility of a candidate itemset, especially in long transaction databases.

Both the UMining and Two-phase are Apriori-like algorithms, they are inadequate in

datasets with long patterns or high dimensions. To solve the problem, we proposed a hybrid top-down/bottom-up search model and a row enumeration-based algorithm, Inter-transaction, to discover all high-utility itemsets from two directions. Under the hybrid model, an existing algorithm such as Two-phase can be used to seek the short high-utility itemsets by starting from the bottom, while Inter-transaction seeks long high-utility itemsets by starting from the top, they complement each other.

Different from inter-transaction association rule (Feng, Li, & Wong, 2001; Feng, Yu, Lu, & Han, 2002; Liu, Feng, & Han, 2000), which aims at discovering correlation between transactions, our Inter-transaction algorithm aims at discovering long high-utility itemsets. It is based on the characteristic that there are few common items between or among long transactions, which means that the intersection of multiple long transactions is usually very short. In a high-dimensional data environment, the characteristic is especially obvious. By intersecting relevant transactions, Inter-transaction can identify long itemsets directly, without extending short itemsets step by step. In addition, new pruning strategies were developed to cut down the search space and an optimization technique was adopted to improve the performance of the intersection of transactions. This paper emphasizes on the introduction of Inter-transaction algorithm.

The remainder of the paper is organized as follows: Section 2 formally defines relevant terms and notations; Section 3 presents the new algorithm; Section 4 introduces an optimization technique. Experimental results are presented in Section 5 and we summarize our work in Section 6.

## DEFINITIONS

This section presents definitions and terminology used in the description of the algorithm. Let  $I = \{i_1, i_2, \dots, i_m\}$  be a set of items,  $T = \{T_1, T_2, \dots, T_n\}$  be a transaction database. Each transaction  $T_q$  in database  $T$  ( $T_q \in T$ ) is a subset of  $I$ , i.e.,  $T_q \subseteq I$ .

*I*. To simplify a notation, we sometimes write a set  $\{i_1, i_2, \dots, i_k\}$  as  $i_1 i_2 \dots i_k$ . Adapting from the notations described in (Yao et al., 2006; Liu et al., 2005; Savasere, Omiecinsky, & Navathe, 1995), we have following definitions:

**Definition 1. The transaction utility of item  $x$  in transaction  $T_q$** , denoted  $u(x, T_q)$ , is the utility brought on by item  $x$  when transaction  $T_q$  occurs. Take the example from Table 1,  $u(A, T_1)=0$ ,  $u(A, T_2)=2$ .

**Definition 2. The transaction utility of itemset  $X$  in transaction  $T_q$** , denoted  $u(X, T_q)$ , is the sum of the transaction utility of item  $x$  contained in  $X$ , i.e.,

$$u(X, T_q) = \sum_{x \in X} u(x, T_q) \quad (1)$$

For example, in Table 1,  $u(AB, T_2) = u(A, T_2) + u(B, T_2) = 2 + 3 = 5$ ,  $u(ABC, T_3) = u(A, T_3) + u(B, T_3) + u(C, T_3) = 4 + 5 + 8 = 17$ . When  $X = T_q$ , we refer to  $u(X, T_q)$  as the **utility of transaction  $T_q$** , denoted  $u(T_q, T_q)$ .

**Definition 3. The partition utility of itemset  $X$  in partition  $P_i$** , denoted  $u(X, P_i)$ , is the sum of the transaction utility of itemset  $X$  in partition  $P_i$  i.e.,

$$u(X, P_i) = \sum_{T_q \in P_i \wedge X \subseteq T_q} u(X, T_q) \quad (2)$$

For more details about partitions, refer to (Savasere et al., 1995).

**Definition 4. The utility of  $X$  in database  $T$** , denoted  $u(X)$ , is the sum of the transaction utility of itemset  $X$  in database  $T$ , i.e.,

$$u(X) = \sum_{X \subseteq T_q \wedge T_q \in P_i \wedge P_i \subseteq T} u(X, P_i) = \sum_{T_q \in T \wedge X \subseteq T_q} u(X, T_q) \quad (3)$$

Examples can be seen in Section 1.

**Definition 5. Transaction identifier list**, denoted **tidlist**, is a set of transaction identifiers.

We define an **intersection transaction**, denoted  $T_{tidlist}$ , as an itemset obtained from the intersection of transactions listed in a **tidlist**. For example, let  $T_1 = ABDFHILM$ ,  $T_2 = ADFGJKOP$ ,  $T_3 = ADFMNOQP$ , then one of the **tidlists** is  $\{1, 2, 3\}$ , and the corresponding intersection transaction  $T_{\{1, 2, 3\}} = T_1 \cap T_2 \cap T_3 = ADF$ . If  $|tidlist|=k$  ( $1 \leq k \leq n$ ,  $n$  is the number of transactions), we refer to the  $T_{tidlist}$  as a  **$k$ -intersection transaction**. A  **$k$ -intersection transaction** is the intersection of  $k$  individual transactions. It can also be regarded as the intersection of two other intersection transactions. For example, the 4-intersection transaction  $T_{\{1, 3, 5, 6\}} = T_1 \cap T_3 \cap T_5 \cap T_6 = T_{\{1, 3\}} \cap T_{\{5, 6\}} = T_{\{1, 5\}} \cap T_{\{3, 6\}}$ , and so on. When  $k=1$ ,  $T_{tidlist}$  is an individual transaction.

Both  **$k$ -intersection transaction** and  **$k$ -itemset** are a set of items, but there are some differences between them. For example, a  **$k$ -itemset** is an itemset with  $k$  items. The term does not tell us any information about its support  $s(T_{tidlist})$  (i.e., the number of transactions containing the itemset); on the contrary, a  **$k$ -intersection transaction** does not tell us how many items the itemset has, but it tells us that the itemset stems from the intersection of  $k$  transactions, with support no less than  $k$ . To distinguish the difference between  $k$  and  $s(T_{tidlist})$ , we refer to  $k$  as the **current support** of  **$k$ -intersection transaction  $T_{tidlist}$** . Obviously,  $k \leq s(T_{tidlist})$ .

For any  $T_{tidlist}$ , we can regard the transactions listed in the **tidlist** as a partition, denoted  $P_{tidlist}$ . According to definitions, the current support of  **$k$ -intersection transaction  $T_{tidlist}$**  should be the number of transactions contained in  $P_{tidlist}$ , denoted  $s(T_{tidlist}, P_{tidlist})$ . We also refer to the corresponding partition utility of the  $T_{tidlist}$  as the **current utility** of  $T_{tidlist}$  under current support (or under current **tidlist**). If  $u(T_{tidlist})$  stands for the utility of  $T_{tidlist}$  and  $u(T_{tidlist}, P_{tidlist})$  for the current utility of  $T_{tidlist}$  according to equation (2) and equation (3),  $u(T_{tidlist}, P_{tidlist})$  should be less than  $u(T_{tidlist})$ . If  $s(T_{tidlist}, P_{tidlist}) = s(T_{tidlist})$ , the corresponding **tidlist** is called **maximal tidlist**, and  $u(T_{tidlist}, P_{tidlist}) = u(T_{tidlist})$  holds. Actually, the meaning of the term “maximal **tidlist**” corresponds to the feature support set in (Pang et al., 2003).



We define a long transaction/itemset/pattern as the transaction/itemset/pattern that includes more than *minlen* items. *Minlen* is a user-defined value. Otherwise, it is called a short transaction/itemset/pattern. Likewise, we define a **high-utility itemset** as the itemset with a utility value higher than a user-specified threshold, i.e., *minutil*. If an itemset is a high-utility itemset, we say the itemset is high, otherwise, the itemset is low. The goal of utility-based ARM is to find all high-utility itemsets. We also define a **long high-utility itemset** as the high-utility itemset with more than *minlen* items.

A local (long) high-utility itemset is a (long) itemset in partition  $p_i$  with partition utility value higher than the **local utility threshold**  $minutil/N$ ,  $N$  is the partition number. A (long) high-utility itemset is also called a global (long) high-utility itemset.

## INTER-TRANSACTION ALGORITHM

This section describes Inter-transaction algorithm. As we know, each itemset is determined either by a transaction or by a group of transactions listed in a *tidlist*. If we let any two transactions intersect each other ( $|tidlist|=2$ ), we can obtain all itemsets (2-intersection transaction) with support no less than two. Likewise, we can obtain all itemsets ( $k$ -intersection transaction) with support no less than  $k$  by intersecting any  $k$  transactions ( $|tidlist|=k$ ,  $1 \leq k \leq n$ ,  $n$  is the number of transactions). Theoretically, by intersecting transactions, we can obtain all itemsets along with corresponding supports and utility values. Like CARPENTER, Inter-transaction is based on row enumeration.

### Partition Method

This subsection presents the necessity and correctness of partition method. Suppose  $n$  is the number of transactions, there will be  $2^n$  combinations of transactions at the worst situation. As the number of rows grows, the explosive growth

of the combination of rows causes the performance of row-enumeration methods decrease dramatically. In a real database,  $n$  can easily reach to several millions, and enumerating all the  $2^n$  intersection transactions is not feasible. To solve the problem, Inter-transaction adopts a partition method to divide a database into multiple partitions, with each partition containing a fitting number of transactions. In the first scan of a database, Inter-transaction finds all local long high-utility itemsets from every partition, and then these local long high-utility itemsets are merged to generate a set of potential long high-utility itemsets. In the second scan of the database, the actual utility and support for these itemsets are computed and global high-utility itemsets are identified. The whole process is just like the one described in (Savasere et al., 1995). The correctness of the partition method is guaranteed by Theorem 1:

**Theorem 1.** Suppose  $T$  is a transaction database,  $P = \{P_1, P_2, \dots, P_N\}$  is a set of partitions of

$$T \left( \bigcup_{i=1}^N P_i = T, P_u \cap P_v = \emptyset, u \neq v \right).$$

If  $X \subseteq I$  is a high-utility itemset, it will appear as a local high-utility itemset in at least one of the partitions.

**Proof.** Let  $X$  be a high-utility itemset, then  $u(X) \geq minutil$ . Divide database  $T$  into  $N$  partitions, then  $X$  may fall into  $M$  partitions ( $1 \leq M \leq N$ ). Assume  $B = \max [u(X, P_i)]$  denotes the biggest partition utility value of  $X$  in all partitions, By definition 4, we have:

$$u(X) = \sum_{X \subseteq T_q \wedge T_q \in P_i \wedge P_i \subseteq T} u(X, P_i) \leq MB \quad (4)$$

If  $B \leq minutil/N$ , then  $u(X) \leq M * minutil/N \leq minutil$ . This is a contradiction.  $\square$

Let  $u$  be total utility value, and coefficient  $\alpha$  be the minimum acceptable ratio of the utility value of an itemset to the total utility value in the database. That is to say,  $minutil = \alpha * u$ .

Suppose we divide the database  $T$  into  $N$  partitions, the local utility threshold ( $minutil/N = a*u/N$ ) should be far larger than the average transaction utility ( $u/n$ ), denoted as  $a*u/N \gg u/n$ . Otherwise, a large number of local high-utility itemsets would be generated. Let  $S$  be the size of partitions, we have:

$$S = n/N \gg 1/a \quad (5)$$

Inequation (5) contradicts the goal of the partition method (reducing the number of transactions in a partition). Experiments (in Figure 8) show that it is applicable for  $S$  to be between  $5/a$  and  $10/a$  in the context of our datasets.

## Task Decomposing

This subsection explains why we decompose the mining task. If the partition number is  $N$ , the average size of partitions will be  $n/N$  (a positive integer), and the total number of potential intersection transactions becomes  $N2^{n/N}$ . When  $n$  is too large, enumerating  $N2^{n/N}$  intersection transactions is still not feasible. The partition method is insufficient in reducing the search space.

Given a proper length threshold  $minlen$ , most of the patterns in a database belong to the category of short itemsets. In other words, long patterns are relatively "sparse" compared with short patterns. Although the number of long patterns is usually much smaller than that of short patterns, it is usually true for these long patterns to cost most of the resources in finding all high-utility (or frequent) itemsets when a down-top method is adopted, since a long pattern always means a lot of short patterns have to be handled ahead. On the other hand, there are few common items between or among long transactions, which means the intersection of multiple long transactions, i.e., intersection transaction is usually very short. By intersecting transactions, we can obtain long itemsets directly, without extending short itemsets step by step. On the contrary, short itemsets are relatively dense; the overhead of enumerating all short intersection transactions is too high.

Based on the different features, it is reasonable for us to decompose the mining task into two subtasks (discovering long patterns and short patterns), so that we can choose proper algorithms to solve them separately.

In the process of identifying long high-utility itemsets, we can filter out all short (intersection) transactions. The rationale behind this method is that short transactions have no effect on the support or utility of long patterns/itemsets, and the intersection of a short transaction with another transaction must be short. Now that the intersection of two long transactions is usually very short, a large number of intersection transactions can be pruned out in time.

## The Algorithm

The subsection describes the Inter-transaction algorithm. Based on above discussions, the new algorithm can be described in Figure 1.

The algorithm is very similar to the partition algorithm (Savasere et al., 1995), but there are two important differences between them. One is that in the partition algorithm, the size of partitions is chosen in terms of the main memory size, such that at least those itemsets and other information that is used for generating candidates can fit in the main memory, whereas Inter-transaction seeks balance between keeping a higher local utility threshold and reducing the number of transactions in a partition: given transaction number  $n$  and  $minutil$ , if partition number  $N$  is small, the local utility threshold ( $minutil/N$ ) will be large (we hope so), but the number of transactions in a partition ( $n/N$ ) will also be large (we want to avoid), and vice versa. The other difference is that in the partition algorithm a certain algorithm such as Apriori is used to generate local frequent itemsets of all length, whereas Inter-transaction discovers only local long high-utility itemsets via enumerating intersection transactions.

In order to compute the current support and current utility of an intersection transaction, a *tidlist* is used to record which transactions are involved in the intersection transaction. If  $T_{tidlist}$  represents the transaction identifier list as-

Figure 1. The inter-transaction algorithm

**Name:** Inter-transaction  
**Input:** A database  $T$ ,  $minutil$ ,  $minlen$   
**Output:** All long high-utility itemsets.

1. Divide  $T$  into  $N$  equal partitions; //  $N$ =number of partitions
2. For ( $i=1$ ;  $i \leq N$ ,  $i++$ ) {
3.     Read in partition  $P_i$ ;
4.     Call subroutine Gen-LHU-Itemsets to obtain all local long high-utility itemsets in  $P_i$ , generating a set of potential long high-utility itemsets  $C^G$ ;
5.     }
6. For ( $i=1$ ;  $i \leq N$ ,  $i++$ ) {
7.     Read in partition  $P_i$ ; // second scan of the data
8.     For each candidate  $c \in C^G$ , compute the utility of  $c$  in terms of equation (3), along with its support;
9.     }
10. If itemset  $c$  is high, output its utility and support.

sociated with  $T_{tidlist}$ , let  $tidlist = tidlist1 \cup tidlist2$  ( $tidlist1 \neq tidlist2$ ), according to relevant definitions, we have:

$$\begin{aligned} T_{tidlist} &= T_{tidlist1} \cup T_{tidlist2} \\ &= T_{tidlist1} \cap T_{tidlist2} \end{aligned} \quad (6)$$

$$\begin{aligned} T_{tidlist}.tidlist &= tidlist \\ &= tidlist1 \cup tidlist2 \end{aligned} \quad (7)$$

$$\begin{aligned} s(T_{tidlist}, P_{tidlist}) &= |T_{tidlist}.tidlist| \\ &= |tidlist1 \cup tidlist2| \end{aligned} \quad (8)$$

$$\begin{aligned} u(T_{tidlist}, P_{tidlist}) &= \sum_{T_q \in P_{tidlist} \wedge T_{tidlist} \subseteq T_q} u(T_{tidlist}, T_q) \\ &= \sum_{T_q \in P_{tidlist}} (u(T_{tidlist}, T_q)) \end{aligned} \quad (9)$$

In equation (9), the condition  $T_{tidlist} \subseteq T_q$  is always met in partition  $P_{tidlist}$ . If  $X$  is a sub-itemset of  $T_{tidlist}$  ( $X \subseteq T_{tidlist}$ ), we can use equation (10) to compute the current utility of  $X$  under the current  $tidlist$ :

$$\begin{aligned} u(X, P_{tidlist}) &= \sum_{T_q \in P_{tidlist} \wedge X \subseteq T_q} u(X, T_q) \\ &= \sum_{T_q \in P_{tidlist}} u(X, T_q) \end{aligned} \quad (10)$$

Both equation (9) and equation (10) stem from equation (2). When the number of transactions in  $P_{tidlist}$  is one, i.e.,  $|P_{tidlist}|=1$ ,  $T_{tidlist}$  denotes an individual transaction, and equation (1) can be regarded as a special form of equation (10). Likewise, when  $|P_{tidlist}|=n$ , i.e.,  $P_{tidlist}=T$ , equation (3) is also a special form of equation (2). For simplicity, in later sections of the paper, we use equation (10) to compute the current utility of an itemset.

Gen-LHU-Itemsets is responsible for generating all local long high-utility itemsets in a partition. To achieve the goal, the subroutine computes the intersection of any two long (intersection) transactions, and then decides whether to call subroutine Mine-Single-Trans to mine every long intersection transaction obtained. The process is repeated until no long intersection transaction is generated. This method guarantees that all the maximal  $tidlists$  of all long intersection transactions can be discovered and finally all local long high-utility itemsets in the partition can be identified. Since the intersection of two transactions is not longer than any of the two transactions, the method has a good convergence. Gen-LHU-Itemsets can be described in Figure 2.

Subroutine Mine-Single-Trans (described in Figure 3) tries to discover all local long high-utility itemsets that an intersection transaction

Figure 2. The Gen-LHU-itemsets subroutine

**Name:** Gen-LHU-Itemsets

**Input:** A partition  $P_i$ ,  $minutil$ ,  $minlen$

**Output:** All local long high-utility itemsets in  $P_i$

1. Take a partition  $P_i$  and calculate the current utility of each long (intersection) transaction  $T_{tidlist}$  ( $T_{tidlist}$  can also be an individual transaction, i.e.,  $|P_{tidlist}|=1$ ) independently according to equation (10). If  $u(T_{tidlist}) \geq minutil/N$ , put  $T_{tidlist}$  into the set of potential long high-utility itemsets:  $C^G = C^G \cup T_{tidlist}$ , and then call subroutine Mine-Single-Trans to identify all local long high-utility itemsets that  $T_{tidlist}$  contains;
2. Perform all the intersections of any two long (intersection) transactions;
3. If the number of long (intersection) transactions is not larger than one, the subroutine ends;
4. Filter out all the short intersection transactions;
5. Check all the long intersection transactions. If  $T_{tidlist1} = T_{tidlist2}$ , merge the repetitious intersection transactions into a single one, i.e.,  $T_{tidlist}$  such that  $T_{tidlist1} = T_{tidlist2}$ ,  $tidlist = tidlist1 \cup tidlist2$ . All the long intersection transactions can form a new partition, go to step 1;

$T_{tidlist}$  contains. First, the subroutine chooses a sorting algorithm to sort the items in descending order by utility values so that we can build multiple monotone decreasing sequences of itemsets. Then, in each monotone decreasing sequence, we build and test long high-utility itemsets by utility values from top to bottom. In this way, a large number of low-utility itemsets can be pruned off.

By replacing  $t_{k-j}$  with  $t_r$ , we can obtain a monotone decreasing sequence of  $k$ -itemsets (from step 5 to step 7). For example, if  $j=1$ , by replacing  $t_{k-1}$  with  $t_k, t_{k+1}, \dots, t_{L-1}$  respectively, we can obtain following monotone decreasing sequence of  $k$ -itemsets:

$$\begin{aligned} & t_0 t_1 t_2 \dots t_{k-2} t_k; \\ & t_0 t_1 t_2 \dots t_{k-2} t_{k+1}; \\ & t_0 t_1 t_2 \dots t_{k-2} t_{k+2}; \\ & \dots \\ & t_0 t_1 t_2 \dots t_{k-2} t_{L-1}. \end{aligned}$$

Obviously,  $u(t_0 t_1 t_2 \dots t_{k-2} t_{k+u} T_{tidlist}) \geq u(t_0 t_1 t_2 \dots t_{k-2} t_{k+v} T_{tidlist})$  ( $0 \leq v$ ). When  $j$  varies from one to  $k$ , multiple monotone decreasing sequences can be generated (step 4 to step 9). For example, let  $j=2$ , by replacing  $t_{k-2}$  with  $t_k, t_{k+1}, \dots, t_{L-1}$  respectively, we can obtain another monotone decreasing sequence:

$$\begin{aligned} & t_0 t_1 t_2 \dots t_{k-3} t_{k-1} t_k; \\ & t_0 t_1 t_2 \dots t_{k-3} t_{k-1} t_{k+1}; \\ & t_0 t_1 t_2 \dots t_{k-3} t_{k-1} t_{k+2}; \end{aligned}$$

$$\dots \\ t_0 t_1 t_2 \dots t_{k-3} t_{k-1} t_{L-1}.$$

In subroutine Mine-Single-Trans, the following pruning strategies are used:

**Strategy 1:** if no  $k$ -itemset is high, all  $(k-j)$ -itemsets must be low and thus can be pruned off ( $1 \leq j \leq k$ ), subroutine ends (step 3).

**Strategy 2:** in each monotone decreasing sequence, if a certain  $k$ -itemset is low, subsequent  $k$ -itemsets in the same monotone decreasing sequence must be low (step 6).

**Strategy 3:** if  $X' = t_0 t_1 t_2, \dots, t_{k-j-1} t_{k-j+1} \dots t_{k-2} t_{k-1} t_k$  is low, all the  $k$ -itemsets in subsequent monotone decreasing sequences must be low (step 8).

In a monotone decreasing sequence, strategy 2 is easy to understand. To understand strategy 3, consider  $X'' = t_0 t_1 t_2, \dots, t_{k-i-1} t_{k-i+1} \dots t_{k-2} t_{k-1} t_k$ , which is obtained by replacing  $t_{k-i}$  with  $t_k$ , the first (and thus the largest) one in a monotone decreasing sequence. When  $i > j$ ,  $u(X'') \geq u(X')$ . So if  $X'$  is low,  $X''$  must be low. For strategy 1, we have following theorem:

**Theorem 2. (Current Utility Upward Closure Property)** Given an intersection transaction  $T_{tidlist}$  if an itemset  $X \subseteq T_{tidlist}$  is high, all superset of  $X$  must be high. In other words, under the

Figure 3. The mine-single-trans subroutine

**Name:** Mine-Single-Trans  
**Input:**  $T_{tidlist}$ ,  $minutil$ ,  $minlen$  //  $T_{tidlist}$  can also be an individual transaction  
**Output:** All local long high-utility itemsets in  $T_{tidlist}$   
**Method:**

- Sort the (intersection) transaction  $T_{tidlist}$  decreasingly by its utility value:  $T_{tidlist} = t_0 t_1 t_2 \dots t_{k-1} t_k \dots t_{L-1}$ , such that  $u(t_i, P_{tidlist}) \geq u(t_j, P_{tidlist}) (i \leq j)$ ;
- For ( $k=L-1$ ;  $k \geq minlen$ ;  $k--$ ) { //  $L$  is the length of  $T_{tidlist}$
- Select the top left  $k$  items to build a  $k$ -itemset  $X$ , i.e.,  $X = t_0 t_1 t_2 \dots t_{k-1}$ . The utility of  $X$  is the largest among all  $k$ -itemsets. Compute  $u(X, P_{tidlist})$  in terms of equation (10), if  $u(X, P_{tidlist}) \geq minutil/N$ , add  $X$  into  $C^G$ , go to next step. Otherwise, the subroutine ends;
- For ( $j=1$ ;  $j \leq k$ ;  $j++$ ) {
- For ( $r=k$ ;  $r \leq L-1$ ;  $r++$ ) {
- Replace  $t_{k-j}$  ( $1 \leq j \leq k$ ) in  $X$  with  $t_r$ , obtaining  $X' = t_0 t_1 \dots t_{k-j-1} t_{k-j+1} \dots t_{k-2} t_{k-1} t_r$ . If  $X'$  is high, add  $X'$  into  $C^G$ , otherwise, exit the loop;
- }
- For any  $j \in [1, k]$ , if  $X' = t_0 t_1 t_2 \dots t_{k-j-1} t_{k-j+1} \dots t_{k-2} t_{k-1} t_k$  (replacing  $t_{k-j}$  with  $t_k$ ) is low, stop building and testing all  $k$ -itemsets that  $T_{tidlist}$  contains;
- }
- }

same current support, if an itemset is low, all its subsets must be low.

**Proof.** Suppose  $I'$  and  $I''$  are two itemset, satisfying  $I' \subset I'' \subseteq T_{tidlist}$ . All the transactions listed in  $tidlist$  form a partition  $P_{tidlist}$ , and every transaction in  $P_{tidlist}$  contains both  $I'$  and  $I''$ . That's to say,  $I'$  and  $I''$  have the same current support. Suppose the current support is  $r$ , let  $I'' = I' + X$  ( $X \neq \emptyset$ ), and  $tidlist = \{1, 2, \dots, r\}$ , by definition 3, we have:

$$u(I', P_{tidlist}) = \sum_{T_q \in P_{tidlist} \wedge I' \subseteq T_q} u(I', T_q) = \sum_{q=1}^r u(I', T_q)$$

Likewise, we have

$$u(I'', P_{tidlist}) = \sum_{T_q \in P_{tidlist} \wedge I'' \subseteq T_q} u(I'', T_q) = \sum_{q=1}^r u(I'', T_q)$$

$$= \sum_{q=1}^r u((I' + X), T_q) = \sum_{q=1}^r u(I', T_q) + \sum_{q=1}^r u(X, T_q)$$

Since  $\sum_{q=1}^r u(X, T_q) > 0$ , then we have:

$$u(I'', P_{tidlist}) > u(I', P_{tidlist}).$$

So if  $I'$  is high,  $I''$  must be high.  $\square$

The following example can show how the subroutine works. Suppose there are three transactions  $T_1$ ,  $T_3$ , and  $T_7$ , their intersection is equal to  $ABCDEF$ , i.e.,  $T_{\{1,3,7\}} = ABCDEF$ , and the corresponding utility values can be seen in Table 2.

After sorting,  $T_{\{1,3,7\}}$  can be expressed as  $ACBEFD$ , with item utility values decreasing gradually. Here the length of  $T_{\{1,3,7\}}$  is six, i.e.,  $L=6$ . If  $minutil=18$ ,  $minlen=3$ , itemsets will be examined in the order shown in Table 3.

Four-itemsets  $ACBE$ ,  $ACBF$  and  $ACBD$  are in the same monotone decreasing sequence, which can be generated by replacing  $E$  of  $ACBE$  with  $F$  and  $D$  respectively. Since  $ACBF$  is low, there is no need to build and test  $ACBD$ .

Table 2. Three transactions and the utility value of each item

	A	B	C	D	E	F
$T_1$	2	1	1	0.3	1	0.5
$T_3$	3	1	2	0.3	1	0.5
$T_7$	1	2	2	0.4	1	1
Partition utility	6	4	5	1	3	2

Table 3. The process of calculating utility values of itemsets

Itemset Utility	Results	Comments
$u(ACBEF, P_{\{1,3,7\}}) = 20$	<i>ACBEF</i> is high	The largest 5-itemset consisting of the top left 5-items
$u(ACBED, P_{\{1,3,7\}}) = 19$	<i>ACBED</i> is high	obtained by replacing <i>F</i> of <i>ACBEF</i> with <i>D</i>
$u(ACBFD, P_{\{1,3,7\}}) = 18$	<i>ACBFD</i> is high	obtained by replacing <i>E</i> of <i>ACBEF</i> with <i>D</i>
$u(ACEFD, P_{\{1,3,7\}}) = 17$	stop finding 5-itemsets	obtained by replacing <i>B</i> of <i>ACBEF</i> with <i>D</i>
$u(ACBE, P_{\{1,3,7\}}) = 18$	<i>ACBE</i> is high	The largest 4-itemset consisting of the top left 4-items
$u(ACBF, P_{\{1,3,7\}}) = 17$	stop finding 4-itemsets	obtained by replacing <i>E</i> of <i>ACBE</i> with <i>F</i>
$u(ACB, P_{\{1,3,7\}}) = 15$	Algorithm end	The largest 3-itemset consisting of the top left 3-items

Although sorting an (intersection) transaction is costly, according to step 1 of subroutine Gen-LHU-Itemsets, only a small number of (intersection) transactions need to call the subroutine Mine-Single-Trans. So this method will not cause high computational cost.

## OPTIMIZATION TECHNIQUE

Generally, a database can usually be implemented in HIV, HIL, VTV and VTL format (Shenoy, Haritsa, Sundarshan, Bhalotia, Bawa, & Shah 2000; Zaki & Gouda, 2003). If we express each transaction in HIV format, an intersection transaction can be obtained from the intersection of bit-vectors. Although the bitwise logical (And) operation is well supported by computer hardware and very efficient, the overall performance of the intersection of two transactions decreases dramatically with the increase of the number of items. For example, if the number of items is eight kilos, we have to use one kilo bytes (8K bits) to express each transaction. In order to perform the intersection of two transactions, eight kilo bit operations are needed and this is not acceptable.

Some optimization techniques such as run-length encoding (RLE), VIPER (Shenoy et al., 2000) and DIFFSET (Zaki & Gouda, 2003) have been proposed to enhance the performance of the intersection of two bit-vectors in vertical mining algorithms. These existing optimizations adopt various compressed for-

ats to store databases. Since data compressing and uncompressing is costly, these methods have limited effect on increasing the speed of the intersection of long bit-vectors. So in our algorithm, we not only refrain from compressing each row in main memory, but also use redundant information to reduce the number of bitwise logical operations.

Besides the HIV format, we also store each transaction in HIL format. Although this method wastes lots of memory, the cost is affordable because the partition method can save lots of memory (every time we read only a partition into a continuous memory address space). HIV format is used to perform the intersection of bit-vectors, while HIL format is used to store redundant information which guide us to choose only necessary bits in a bit-vector to perform bitwise logical (And) operation. If the length of transaction  $T_i$  in HIL format is longer than that of  $T_j$ , we choose  $T_j$  (the shorter transaction) as the benchmark to determine on what bits the bitwise logical operation should be performed: if the  $k$ -th bit in  $T_j$  is one, "And" operation should be performed on the  $k$ -th bit, and the other bits should be set zero directly in the result.

For example, there are three transactions  $T_1$ ,  $T_2$ , and  $T_3$ , and the corresponding data can be seen in Table 4. If we want to compute  $T_{\{1,2,3\}}$ , we can firstly compute  $T_{\{1,2\}}$  by intersecting  $T_1$  with  $T_2$ , and then we compute  $T_{\{1,2,3\}}$  by intersecting  $T_{\{1,2\}}$  with  $T_3$ . In step one, we choose the shorter transaction  $T_2$  as the benchmark and decide bitwise logical (And) operation



should be performed only on the first bit, the fourteenth bit and the fifteenth bit (written in bold Italic in Table 4), other bits should be set at zero. As an intermediate result, we obtain  $T_{\{1,2\}} = \{1,15\}$  (in HIL format). In step two, we choose  $T_{\{1,2\}}$  as the benchmark and decide that bitwise logical (And) operation should be performed only on the first bit and the last bit. We get the final result  $T_{\{1,2,3\}} = \{1,15\}$ . As shown in Table 4, only five bit operations are needed for the whole process.

In this way, the number of bit operations linearly depends only on the length of the short transaction in HIL format. Experiment shows this optimization technique is a key to the success of Inter-transaction.

## EXPERIMENTAL RESULTS

All the experiments were performed on a 2GHz Legend server with 4GB of memory, running windows 2003. The program was coded in Delphi 7.

Seven datasets were used in our experiments; all were generated by IBM quest data generator (retrieved from the web May 10, 2007. [http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data\\_mining/datasets/syndata.html](http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html)). Six of them are T40.I30.D8000K with 0.5K, 1K, 2K, 4K, 8K and 16K items respectively, the seventh is T20.I6.D8000K with 4K items, where T# stands for the average length of transactions, I# for the average length of maxi-

mal potentially large itemsets and D# for the number of transactions. Because the generator only generates the quantity of zero or one for each item in a transaction, we use Delphi function "RandG" to generate random numbers with Gaussian distribution, which mimic the quantity sold of an item in each transaction. The unit profit of each item is defined as item ID%100, where % is a modulus operator. The utility threshold *minutil* is defined as the minimum acceptable ratio of the utility value of an itemset to the total utility value in the database.

Figure 4 presents the scalability of Inter-transaction by increasing the number of transactions from 0.25M (million) to 8M. Experimental results show that our algorithm scales linearly with the number of transactions.

Figure 5 shows the performance when the number of items varies. Different from other algorithms, the performance of Inter-transaction improves as the number of items increases. The reason is that the number of items is directly related to the sparseness of a dataset. The more items there are, the sparser the dataset is, and the shorter the intersection of two transactions may be. That means Inter-transaction can enumerate all long intersection transactions easily in a sparse dataset. From Figure 5 we can observe that Inter-transaction is suitable for those datasets with more than one kilo (1K) items.

In Figure 6, *minlen* (a positive integer) is the minimum length of itemsets that the Inter-transaction can discover within a reasonable time (within 2 hours). It actually determines the task assigned to Inter-transaction. Figure 6 shows that *minlen* decreases as the number of items increases, which means Inter-transaction can complete more mining tasks in a sparse database. The reason is just the same as mentioned above.

Figure 7 shows the execution time of Inter-transaction when the utility threshold varies. Since the number of candidate itemsets decreases as the minimum utility threshold increases, the execution time decreases, correspondingly.

Figure 8 shows that the size of partitions is very important to Inter-transaction. Just as we

Table 4. Three transactions and the process of computing  $T_{\{1,2,3\}}$

	HIV format	HIL format
$T_1$	1011,0000,1100,001	1,3,4,9,10,15
$T_2$	<b>1</b> 000,0000,0000, <b>011</b>	1,14,15
$T_{\{1,2\}}$	<b>1</b> 000,0000,0000, <b>001</b>	1,15
$T_3$	1011,1100,0001,001	1,3,4,5,6,12,15
$T_{\{1,2,3\}}$	1000,0000,0000,001	1,15

Figure 4. Scalability with the number of transactions

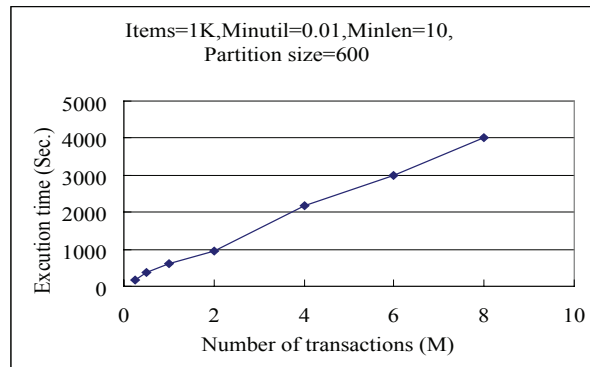


Figure 5. Scalability with the number of items

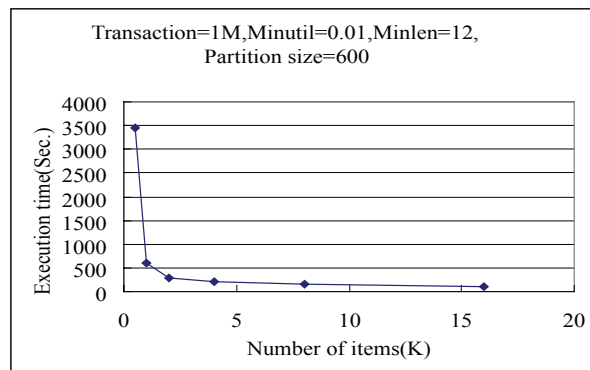


Figure 6. The effect of the number of items on minlen

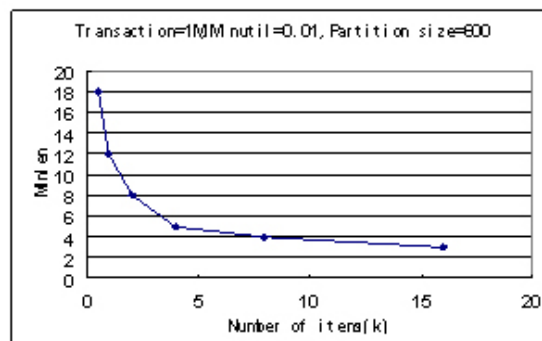


Figure 7. Scalability with utility threshold

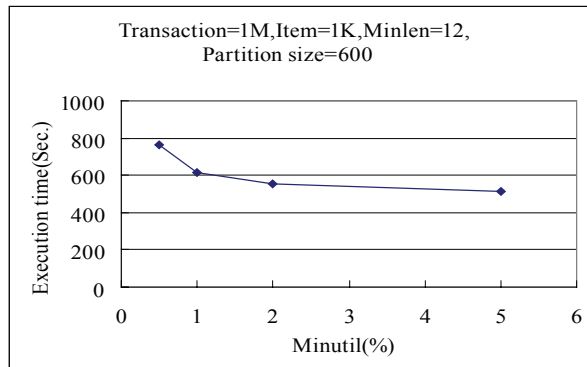
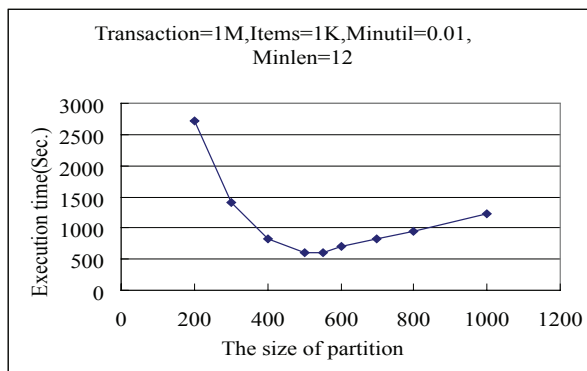


Figure 8. The effect of size of partitions on performance



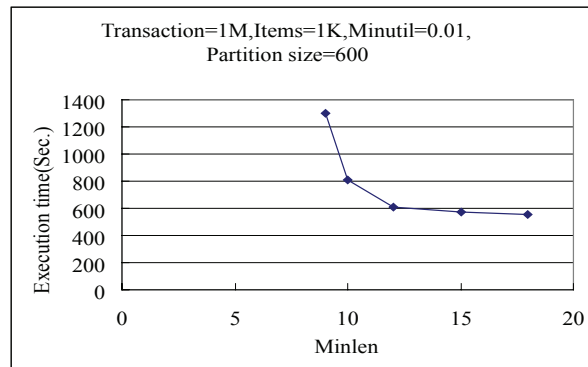
have mentioned above, a partition that is too small or too large will degrade the performance of the algorithm. From Figure 8, we can see the reason why we set the size of partitions to 600 in all other experiments. When *minutil* is divided into half of the initial value, that is, *minutil*/2, we can keep the local utility threshold unchanged by doubling the size of partitions. In this way, large amounts of candidate itemsets can be avoided. This is why our method can work well under a small *minutil*.

Figure 9 shows the effect of *minlen* on the performance of Inter-transaction. As we have mentioned, *minlen* actually assigns mining tasks between Inter-transaction and its cooperator, such as Umining or Two-phase. The larger the *minlen*, the fewer the tasks assigned to Inter-transaction, and the shorter the execution time

needed for Inter-transaction. Although a larger *minlen* always means a shorter execution time for Inter-transaction, more tasks will be left for its cooperator, and the overall performance is not necessarily high. On the other hand, a too small *minlen* does not benefit the overall performance of the hybrid model either. If *minlen* is too small, Inter-transaction has to enumerate too many intersection transactions, the running time increases dramatically. So a proper *minlen* is a key parameter for the hybrid model. Figure 6 indicates that we should choose different *minlen* in terms of the number of items.

To test the total performance of the hybrid model, we choose Two-phase to mine short high-utility itemsets, then compare the performance of the hybrid method (here hybrid method means Inter-transaction + Two-phase) with that of

Figure 9. The effect of minlen on performance



Two-phase. That is, the total running time of the hybrid method is equal to the running time of Inter-transaction (used to find long high-utility itemsets) plus the running time of Two-phase (used to find short high-utility itemsets). The reason for choosing Two-phase is that we are certain that it is the best approach for utility mining by now. Experiments were performed on T20I6D8000K and T40I30D8000K, with one kilo and four kilo items, respectively. *Minlen* is set three for T20I6D8000K and five for T40I30D8000K, corresponding performance curves are illustrated in Figures 10, 11, 12 and 13.

From Figures 10 and 11 we can observe that the hybrid model does not suit datasets

with only short patterns. The reason is that Inter-transaction can not take obvious effect on these datasets. As for those datasets with lots of long patterns or high dimensions, Two-phase has to extend short itemsets step by step to obtain long itemsets, while Inter-transaction can obtain long itemsets directly by intersecting relevant transactions. In this situation, the hybrid method has great advantages over the Two-phase algorithm. Figures 12 and 13 show the hybrid method is not sensitive to utility threshold *minutil*. With *minutil* growing down, the performance of Two-phase decreases rapidly. When *minutil* is less than 0.25, our hybrid method outperform Two-phase even on the datasets with only short patterns.

Figure 10. Scalability with the number of transactions on T40I30D8000

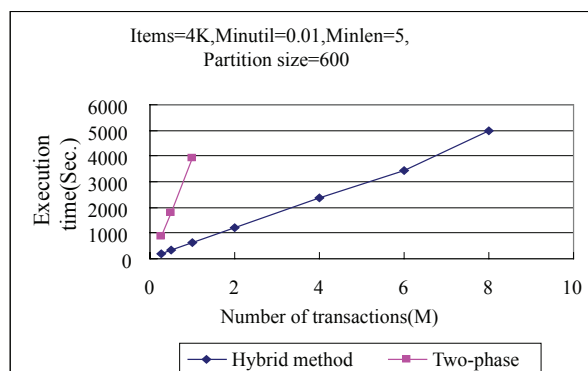


Figure 11. Scalability with the number of transactions on T20I6D8000

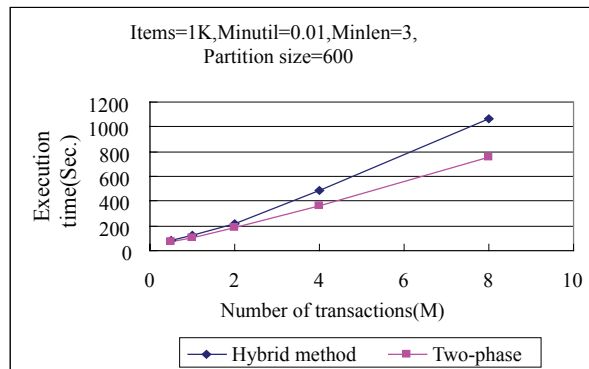


Figure 12. Scalability with utility threshold on T40I30D8000

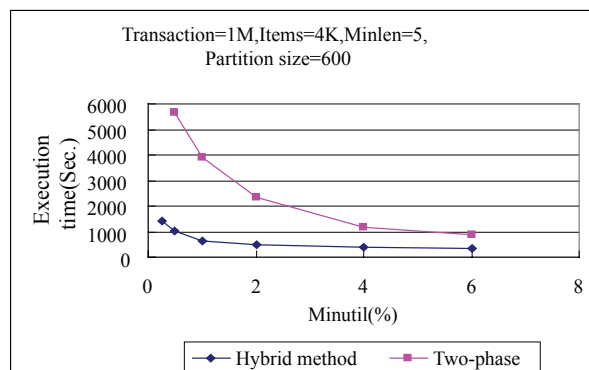
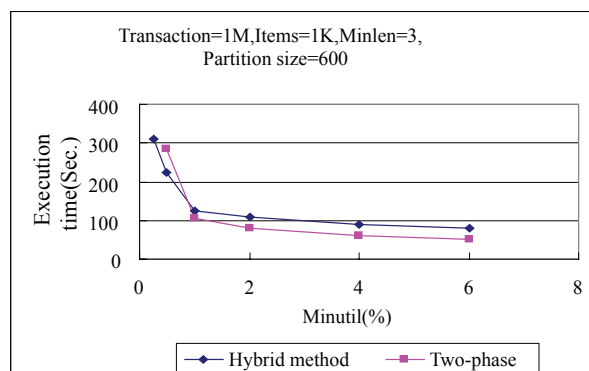


Figure 13. Scalability with utility threshold on T20I6D8000



## CONCLUSION

The paper proposes a hybrid model to discover high-utility itemsets from two directions. The intention of the hybrid model is to decompose a complex problem into two easy subtasks, and then use proper methods to solve them separately.

By integrating the advantages of the partition algorithm and row enumeration algorithms, Inter-transaction can handle large high-dimensional databases efficiently. It scans a database at most twice, and is ideally suited for parallelization. Since inter-transaction is based on the sparseness of long patterns, its performance is affected by the characteristics of the database, including data skew and the number of items. Parameters such as minimum length threshold *minlen* and the size of partitions also affect its performance. In our next plan, we will make an intensive study of how to choose minimum length threshold *minlen* to achieve the best performance for the hybrid model.

## REFERENCES

- Agarwal, R. C., Aggarwal, C. C., & Prasad, V.V.V. (2000). A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 61(3), 350-371.
- Aumann, Y., & Lindell, Y. (2003). A Statistical Theory for Quantitative Association Rules. *Journal of Intelligent Information Systems*, 20(3), 255-283
- Barber, B., & Hamilton, H. J. (2003). Extracting Share Frequent Itemsets with Infrequent Subsets. *Data Mining and Knowledge Discovery*, 7, 153-185.
- Cai, C. H., Fu, A. W. C., Cheng, C. H., & Kong, W. W. (1998). Mining Association Rules with Weighted Items. *Proceedings of the International Database Engineering and Applications Symposium*. Cardiff, UK, (pp. 68-77).
- Chan, R., Yang, Q., & Shen, Y. D. (2003). Mining high utility itemsets. *Proceedings of the 3rd IEEE International Conference on Data Mining*, (pp. 19-26).
- Feng, L., Li, Q., & Wong, A. (2001). Mining Inter-Transactional Association Rules: Generalization and Empirical Evaluation. *Proc. of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, Lecture Notes in Computer Science, Germany, (pp. 31-40).
- Feng, L., Yu, J. X., Lu, H. J., & Han, J. W. (2002). A Template Model for Multidimensional Inter-Transactional Association Rules. *International Journal of Very Large Data Bases*, 11(2), 153-175.
- Han, J. W., Pei, J., Yin, Y., & Mao, R. (2004). Mining Frequent Patterns without Candidate Generation: A Frequent pattern Tree Approach, *Data Mining and Knowledge Discovery*, 8 (1), 53-87.
- Lin, T. Y., Yao, Y. Y., & Louie, E. (2002). Mining Value Added Association rules. *Proceedings of PAKDD*, (pp. 328-333)
- Liu, H., Feng, L., & Han, J. (2000). Beyond intra-transactional association analysis: Mining multidimensional inter-transaction association rules. *ACM Transactions on Information Systems*, 18(4), 423-454.
- Liu, Y., Liao, W. K., & Choudhary, A. (2005). A fast high-utility itemsets mining algorithm. *Proceedings of the First International Workshop on Utility-based Data Mining*. Chicago, Illinois, (pp. 90-99).
- Lu, S. F., Hu, H. P., & Li, F. (2001). Mining weighted association rules. *Intelligent Data Analysis*, 5(3), 211-225. IOS Press.
- Ngan, S. C., Lam, T., Wong, R. C. W., & Fu, A. W. C. (2005). Mining N-most interesting itemsets without support threshold by the COFI-tree. *Journal of Business Intelligence and Data Mining*, 1, 88-106
- Pang, F., Cong, G., Tung, A., Yang, J., & Zaki, M. (2003). Carpenter: Finding closed patterns in long biological datasets. *Proceedings of the SIGKDD '03*. (pp. 637-642).
- Savasere, A., Omiecinski, E. & Navathe, S. (1995). An efficient algorithm for mining association rules in large databases. *Proceedings of 21st Int'l Conf. on Very Large Databases*. Zurich, Switzerland, (pp. 432-444).
- Shen, Y. D., Zhang, Z., & Yang, Q. (2002). Objective-oriented utility-based association mining. *Proceedings of the 2002 IEEE International Conference on Data Mining*. Maebashi City, Japan, (pp. 426-433).



- Shenoy, P., Haritsa, J. R., Sundarshan, S., Bhalotia, G., Bawa, M., & Shah, D. (2000). Turbo-charging Vertical Mining of Large Databases. *Proceedings of ACM SIGMOD International Conference on Management of Data*. Dallas, Texas, (pp. 22-33).
- Tjioe, H.C., & Taniar, D. (2005). Mining association rules in data warehouses. *Journal of Data Warehousing and Mining*, 1(3), 28-62.
- Vroom, V. H. (1964). *Work and Motivation*. New York, John Wiley & Sons.
- Wang, K., Zhou, S., & Han, J. (2002). Profit mining: from patterns to action. *Proceedings of International Conference on Extending Database Technology*. (pp. 70-87).
- Webb, G. I. (2001). Discovering associations with numeric variables. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. San Francisco, California, (pp. 383-388).
- Yao, H., Hamilton, H. J., & Geng, L. (2006). A Unified Framework for Utility Based Measures for Mining Itemsets. *Proceedings of the 2006 International Workshop on Utility-Based Data Mining*. Philadelphia, USA, (pp. 28-37).
- Yao, H., & Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering*, 59, 603 – 626.
- Zaki, M. J., & Gouda, K. (2003). Fast vertical mining using diffsets. In *Proc. of ACM SIGKDD*. Washington DC, (pp. 326-335).

*Guangzhu Yu received his master's degree in computer science from the Yangtze University, Jingzhou, China in 2002. He is currently a doctoral student at the Donghua University (formerly China Textile University), Shanghai, China. His research interests include data mining and network security.*

*Shihuang Shao received his bachelor's degree in electrical engineering from Southeast University, Nanjin, China, in 1960. He was a Visiting Scientist at the University of Maryland, College Park, from 1986 to 1988, and was the Chairman of Donghua University, Shanghai, China, from 1994 to 2001. He is currently a Professor of the same university. His research interests include fuzzy control, neural networks, genetic algorithms, chaos control, and data mining.*

*Bing Luo was born in April 1966. He got his master's degree from Jiangnan Petroleum Institute in 1996 and doctor's degree from Guangdong University of Technology in 2007. Now he is working in Guangdong University of Technology as an associate professor. His research interests include data mining, information acquisition and processing.*

*Xianhui Zeng received his master's degree from the Donghua University, Shanghai, China, in 1999. He is currently an associate professor and a doctoral student at the same University. His research interests include data mining, decision support system and intelligent information processing.*

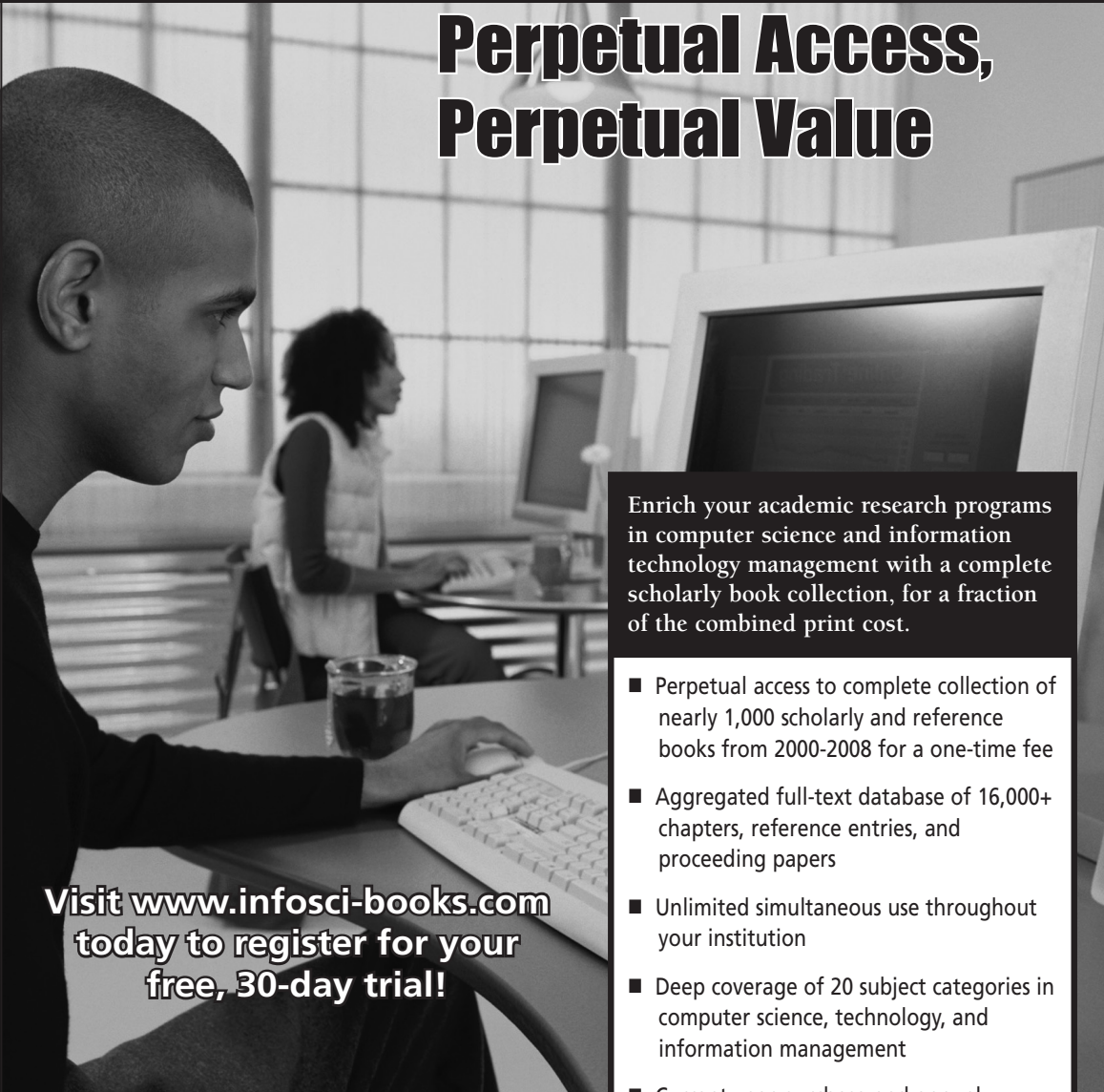


The Industry Leader in Delivering  
Knowledge in Computer Science and  
Information Technology Management

AFFORDABLE | AUTHORITATIVE | COMPREHENSIVE

## INTRODUCING ... InfoSci-Books™

### Perpetual Access, Perpetual Value



Enrich your academic research programs in computer science and information technology management with a complete scholarly book collection, for a fraction of the combined print cost.

- Perpetual access to complete collection of nearly 1,000 scholarly and reference books from 2000-2008 for a one-time fee
- Aggregated full-text database of 16,000+ chapters, reference entries, and proceeding papers
- Unlimited simultaneous use throughout your institution
- Deep coverage of 20 subject categories in computer science, technology, and information management
- Current-year purchase and annual subscription options also available
- Discounts for consortia and other multi-site groups

Visit [www.infosci-books.com](http://www.infosci-books.com)  
today to register for your  
free, 30-day trial!



Order through most subscription  
agents or directly from IGI Global

[www.infosci-books.com](http://www.infosci-books.com)

IGI Global • 701 E. Chocolate Ave., Suite 200 • Hershey, PA 17033-1240 USA  
1-866-342-6657 or 717-533-8845 ext. 100 • Fax: 717-533-8661 • [eresources@igi-global.com](mailto:eresources@igi-global.com)