Universidade do Minho

Escola de Engenharia

Ronnie Cley Oliveira Alves

Aggregation-based Mining Methods: From Single to N-dimensional Data Analysis

October 2007

Universidade do Minho
Escola de Engenharia

Ronnie Cley Oliveira Alves

Aggregation-based Mining Methods: From Single to N-dimensional Data Analysis

Thesis for the degree of Doctor in Informatics

Elaborated under the supervision of
Prof. Dr. Orlando Manuel de Oliveira Belo

October 2007

Dedicated to my parents Raimundo and Rosalda, my wife Ana and my son Pedro Miguel

Acknowledgements

This is one of the best moments in my doctoral program - to publicly acknowledge those who have contributed, in many different ways, to make my success a part of their own.

I am most fortunate to have had Prof. Dr. Orlando Belo as my supervisor. I wish to express my deep gratitude for his guidance and continuous encouragement through these years. He seamlessly encouraged me to be pro-active in learning.

My sincere thanks to Prof. Dr. Pedro Henriques for help me and encouraging me to pursue a research assistant position in the IKF project while waiting for a proper financial fund for my PhD research at the Department of Informatics at the University of Minho. I am indebted to him for his continuous support.

Special thanks to all administrative staff, in particular to Paula Anjo for her administrative assistance and support during my PhD program.

I had the privilege of visiting the University of Illinois at Urbana-Champaign, United States. Prof. Dr. Jiawei Han allowed me to work with the Data Mining Research Group as visiting scholar during the Fall 2006. I had the opportunity to share and implement some of my ideas to the improvement of this research. I am thankful to Dr. Han for his support and hospitality. I greatly benefited from this collaboration. Thanks also to Xiaolei Li, Hector Gonzalez, Dong Xin, and Hong Cheng for being so available to help.

I was also fortunate of visiting the Universidad Pablo de Olavide at Sevilla, Spain. Prof. Dr. Jesus Aguilar gave me the opportunity to work with the Bioinformatics Research Group as visiting scholar during Fall 2007. I am grateful to Professors Jesus Aguilar and Roberto Ruiz for his support.

I have very much enjoyed being a member of the Department of Informatics at the University of Minho. I am grateful to the professors and other fellow students – Pedro Ferreira, Joel Ribeiro, Fabio Costa, Cristina Wanzeller, Nuno Lopes, Mario Beron and Anália Lourenço.

I would like to express my gratitude to Pedro Ferreira e Joel Ribeiro for their partnership on several discussions concerning this thesis, especially in Chapter 3 and Chapter 5.

Special thanks to my Brazilian friends – Giovani Librelotto, Marco Barbosa, Alfranio Tavares and Gustavo Arnold.

I also acknowledge the Portuguese funding agency FCT (Fundação para Ciência e a Tecnologia) of Ministry for Science, Technology and Education of Portugal, for the indispensable financial support under the grant SFRH/BD/17422/2004.

Acknowledgment is due to FCG (Fundação Calouste Gulbenkian) for the vital financial support under grants 81687/2006 and 87749/2007.

Resumo

Mineração de Dados baseados em Agregações: Da análise Unidimensional à análise Multidimensional de Dados

Aplicações nas áreas do comercio electrónico, telecomunicações, varejo, acções, bio informática, entre outras, possuem, quase que de forma natural, uma organização multidimensional e multinivel na informação com que lidam. Vários dos esforços desenvolvidos em acções de pesquisa, particularmente nas áreas de Data Warehousing e OLAP, têm sido realizados para no sentido de permitir a identificação de padrões, ditos interessantes, em diversos níveis de abstracção, usufruindo da organização multidimensional desses dados. A esta estratégia, voltada para a *Análise de Dados Multidimensionais* (ADM), atribui-se vulgarmente a designação de OLAPing. Em termos gerais, a ADM tem como principal objectivo projectar métodos que sejam, ao mesmo tempo, eficientes e efectivos em processos de computação de funções de agregação sobre base de dados de alta dimensionalidade. Estes casos, são um grande desafio para a ADM, particularmente pelas limitações encontradas pelos métodos tradicionais para a computação de agregações. Assim, é vulgar questionar, como desenvolver métodos que possam não somente serem orientados para os dados, mas também serem orientadas para a descoberta de padrões relevantes nestas mesmas bases de dados? Além disto, os dados e os padrões sofrem ao longo do tempo mudanças significativas. Isso levanta outro tipo de questões. Em particular, como apontar estas sensíveis evoluções tomando como partida estruturas multidimensionais? Os *Métodos para Mineração baseado em Agregações* (MMA) resolvem problemas como os descritos anteriormente. Esta tese posiciona-se num ponto um pouco mais à frente na pesquisa nos domínios do OLAPing. Neste sentido, foram desenvolvidos diversos métodos para mineração baseado em agregações para ADM de acordo com a dimensionalidade e também para com a análise a ser dirigida sobre os dados. Complementarmente, desenvolveram-se algumas novas medidas de interesse para auxiliar o processo de identificação de agregações interessantes durante a computação de estruturas multidimensionais.

Abstract

Aggregation-based Mining Methods: From Single to N-dimensional Data Analysis

In many applications, data contains structured information that is multidimensional and multilevel in nature, such as the ones in areas like e-commerce, telecommunications, retail, stocks, or bioinformatics. Since the last decade, we have been facing several research efforts on Data Warehousing and OLAP to get better view of multidimensional data, allowing a data-driven search for interesting patterns at several levels of data abstraction. This strategy of *Multidimensional Data Analysis* (MDA) on multidimensional databases is also called OLAPing. MDA relies on effective and efficient computation of aggregating functions on high dimensional databases. MDA on higher dimensional databases is not a trivial task, given the limitation of the most known aggregation-based algorithms. So, how to enhance this data-driven search with discovery-driven features smoothing the curse-of-dimensionality problem? Besides, data as well as patterns evolve over time-to-time. Thus, how to highlight those significant changes? *Aggregation-based Mining Methods* (AMM) takes its place helping to handle those previous issues. In summary, AMM attempts to combine ideas of aggregating and mining functions to get better mechanisms for practical and effective MDA. The work presented in this thesis lies outside of traditional work on OLAPing. These previous discussions guided us to devise several methods to achieve *Multidimensional Data Mining* (MDM) by applying aggregation-based mining methods. These methods cover several levels of data summarizations, from single to n-dimensional, as well as, advanced aggregation-based mining. Apart from these methods, we have also designed new measures of interestingness for evaluating significant aggregations during processing time.

Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

Chapter 1

Introduction

1.1 Aggregation-based Mining Methods

For a long time, organizations have been using decisions support systems to help them to understand and to predict interesting business opportunities over their huge databases. This interesting knowledge is gathered in such way that one can explore different what-if scenarios over the complete set of information available. Those huge databases are well known as *Data Marts* (DM), organizing the information and preserving its multidimensional and multilevel characteristics. OLAP tools have been used widely by DM users for retrieving summarized information, also called multidimensional data cube or just cube through customized cubing algorithms. The kernel of cubing algorithms relies on two main spots, namely, partition strategies and aggregating functions. Cubing can also be delivered from single to n-dimensional DM. The more dimensions one select to cube the more computational resources are needed to afford MDA. Computing high dimensional DM is a challenge task [Sismanis et al. 2002] [Lakshmanan et al. 02] [Feng et al. 04] [Li et al. 04] [Han & Kamber 06] [Alves & Belo 06b] [Alves & Belo 07b].

Since DM evolves, it is also necessary to have the cube updated on a timely basis. Usually, traditional cubing methods compute the cube structure from scratch every time new information is available. As far as we know, almost no one of them support an incremental procedure. Furthermore, traditional cubing approaches suffer from being just data-driven oriented and not discovery-driven ones. In fact, real data application demands both strategies [Sarawagi et al. 98a] [Sarawagi et al. 98b] [Han & Kamber 06].

Other relevant aspect of MDA is the granularity of the information. When exploring cubes either is required detailed or general information. Concept hierarchy modeling comes as a required solution for setting the right level of information, providing an enhanced model for multidimensional and multilevel mining. Rather than using the TID structure like multidimensional association rules do for multilevel mining, the cube structure itself provides a suitable data structure for MDA at several levels of abstraction. However, cubing *per-se* also incurs computational and information management costs. Thus, it is also important to provide a basic infrastructure providing a good tradeoff between aggregation and extraction mechanisms for MDA. Furthermore, handling a large number of dimensions and hierarchy's levels might be quite challenge task for cubing methods. Therefore, bringing out some mining technique into the cubing process is an essential effort to reveal interesting relations on DM [Kamber et al. 97] [Lu et al. 00] [Han et al. 99] [Alves & Belo 07a].

On the other hand, there are also real application scenarios where it is not possible to have a n-dimensional cube either for computational costs or for the application itself. In such cases, one can also make use of the partition-and-aggregation model of cubing methods to summarize a particular dimension of interest with complex aggregating functions and then push mining with constraints. Thus, mining should guide to highlight interesting patterns over the aggregated data. Defining proper levels of aggregation-based mining in MDA is also a challenge problem.

Clearly, MDA requires suitable *Multidimensional Data Mining* (MDM) methods. It is a non-trivial task to decide what level of aggregation-based mining is necessary for revealing interesting spots in DM. In addition, it poses new challenges for novel uses of data cubing and data mining technology.

The work presented in this thesis lies outside of traditional work on OLAP mining. These previous discussions guided us to devise several methods to achieve multidimensional data mining through aggregation-based mining strategies. These methods cover several levels of data summarizations, from single to n-dimensional, as well as, advanced aggregation-based mining. Apart from the methods, we have also designed new measures of interestingness for guiding high dimensional data analysis.

1.2 Thesis Statement

In this work, we investigate the feasibility of achieving MDM by employing aggregation-based mining techniques. The central thesis statement of this research is presented as follows:

Multidimensional data mining, by aggregation-based mining, is to some extent as possible.

In many applications, data contains structured information that is multidimensional and multilevel in nature, such as the ones in areas like e-commerce, telecommunications, retail, stocks or bioinformatics. Since the last decade, we have been facing with several research efforts on Data Warehousing and OLAP to get better view of multidimensional data, allowing a data-driven search for interesting patterns at several levels of data abstraction [Han & Kamber 06]. This strategy of searching patterns on multidimensional databases is also called OLAPing (or data cubing). The basis of MDA relies on effective and efficient computation of aggregating functions. MDA on higher dimensional databases is a quite hard task; given the limitation of the most known cubing algorithms. So, how to enhance this data-driven search with discovery-driven features smoothing the curse-of-dimensionality problem? Besides, data as well as patterns evolve over time-to-time. Thus, how to highlight those significant changes? MDM take its place helping to handle those previous issues. In summary, MDM attempts to combine ideas of aggregating and mining functions to get better mechanisms for multidimensional data analysis.

Three major issues are addressed to support this central statement:

1. It is possible ***to achieve a simple infrastructure for MDM***, i.e., by combining traditional frequent pattern mining methods.
2. It is possible ***to evaluate evolving multidimensional and multilevel patterns*** through providing different levels of aggregation-based mining methods, i.e., from single to n-dimensional mining.
3. It is possible ***to achieve advanced MDA*** by pushing and evaluating measures of interestingness during the processing step, i.e., cubing-based mining of high n-dimensional databases.

1.3 Thesis Contributions

The major contributions of this thesis can be summarized as follows:

- ***A frequent pattern mining strategy closer to RBDMS***. Most of the itemset mining approaches are memory-like and run outside of the database. On the other hand, when dealing with data marts the size of tables is extremely huge for memory copy. In addition, using a pure SQL-like approach is quite inefficient. Actually, those implementations rarely take advantages of database programming. We devised a pattern growth mining approach by means of database programming for finding all frequent itemsets. The main idea is to avoid one-at-a-time record retrieval from the database, saving both the copying and process context switching, expensive joins, and table reconstruction. This strategy is described in Chapter 2.
- ***A hybrid method for mining inter-dimensional patterns***. Classical association rules are by nature intra-transaction based, i.e., the associations occurs among the items within the same transaction. Further, those transactions are usually associated with a dimensional context which is typically ignored. This prevents that patterns occurring along this dimension (i.e. associated context) being discovered. We devised a strategy for the extraction and analysis of inter-dimensional patterns. These patterns allow us to discover relationships that occur among transactions within a dimensional time frame. This strategy consists in the combination of frequent and sequence pattern mining methods. We also have explored a measure called cvd which allow us to reject patterns that presents significant distance variation, and thus, avoiding the representation of misleading patterns. This method is also described in Chapter 2.
- ***A family of aggregation-based single-dimensional mining methods***. We studied the problem of superimposed fraud detection in telecommunication systems. We proposed three aggregation-based single-dimensional mining methods for anomaly detection. These methods rely on a single level of aggregation called signatures. The first method is a signature deviation-based approach while the second one is designed for dynamic clustering analysis. Furthermore, we enhanced this anomaly detection strategy by using a graph-based approach representing the social or community behavior, without losing information, and also providing a rich structure for revealing interesting calling patterns. We used annotated digraphs to represent the call graph network. The dynamics of changes are modeled into the graphs by weighting factors using damping ideas. This representation highlights the fact that more recent CDR contributes more heavily to the graph at time t. Thus, new information is aggregated

in accordance with this representation and evaluated through pre-defined recursive functions. All these methods are described in Chapter 3.

- ***An incremental cube-based mining method for mining multidimensional and multilevel patterns***. The updating processes on real DM applications implies that, the summarization made by cubing methods need to handle, with some incremental and discovery facility, the new information available. Furthermore, mining mechanisms should be integrated into the cubing process to improve the exploration of interesting relations on DM either before, after or while OLAPing. We described a method which is quite competitive w.r.t. other known cubing strategies, also providing an effective cube-based mining strategy for discovering knowledge over evolving DM. We also demonstrate that by exploring correlated cuboids during the discovery process we are able to provide an essential tradeoff between processing time and pattern accuracy. Such strategy is presented in Chapter 4.
- ***Advancing aggregation-based mining methods***. Addressing advancing aggregation-based mining in high n-dimensional datasets requires different kinds of pushing constraints into the aggregation process, since high MDA suffers from the curse of dimensionality problem. We propose two distinct methods for MDA in high dimensional spaces. The first method proposes a new notion for reducing cuboids aggregation by means of computing only the maximal-correlated cuboids cells, and presents a strategy that brings out those cuboids. This strategy is a promising candidate for scalable data cubing, reducing the number of cuboids by at least an order of magnitude or more in comparison with that of closed ones. The second method attempts to find interesting changes in high n-dimensional spaces. Thus, we are interested to explore best cases (Top-K cells) of interesting multidimensional gradients. We reviewed iceberg cubing for complex measures, since it is the basis for mining gradient cells. We also propose a gradient-based cubing strategy to evaluate interesting gradient regions in MDS. Thus, the main challenge is to find maximum gradient regions (MGRs) that maximize the task of mining Top-K gradient cells. Both advanced aggregation-based mining methods are described in Chapter 5.

Our contributions mentioned previously are the major parts of an MDM infrastructure depicted in Figure 1.

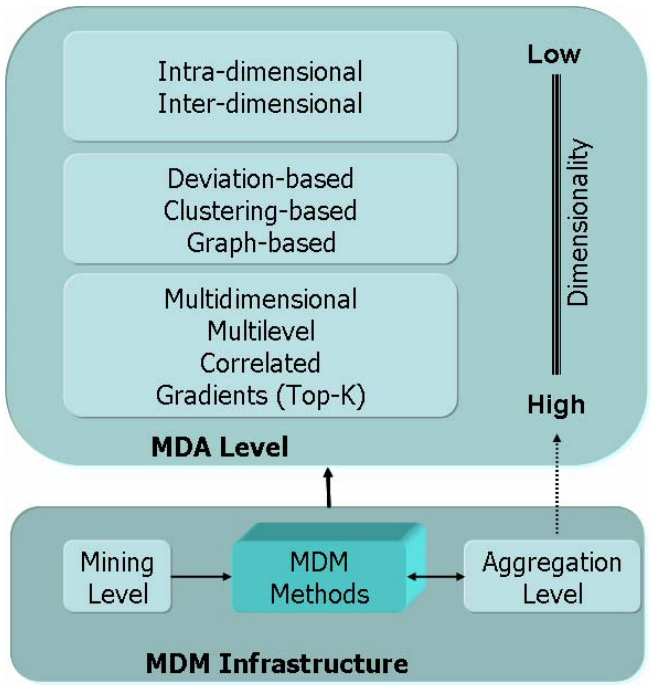


Figure 1. A schematic view of the infrastructure for MDM.

1.4 Research Methodology

This research was conducted in a phased approach, as follows:

- ***Evaluation of prior work***. Research in the area of MDM is still at a very preliminary stage. We have surveyed efforts to address MDM and investigated the related literature. The existing solutions for achieving MDM are reviewed in each chapter according to its level of aggregation and application, i.e., single- to n-dimensional mining. Chapters in this thesis are also self-contained, presenting related background, summaries and final discussions.
- ***Conception and design of new aggregation-based mining methods***. We devised new methods to achieve MDM through aggregation-based mining strategies. These methods cover several levels of data summarization, from single-dimensional to n-dimensional, as well as aggregation-based mining. Apart from the methods, we have

also devised new measures of interestingness for guiding multidimensional data analysis.

- ***Implementation and test***. We have implemented and tested both existing and proposed MDM methods. We also applied our methods to real and synthetically generated datasets to evaluate its effectiveness and to study their performance.
- ***Evaluation***. We evaluated our methods for different aggregation-level scenarios. For each level of aggregation we carried out several studies by using distinct datasets. These studies, implemented in a common platform, were used to find out proper level of aggregation-based mining for each type of application. For newly designed aggregation-based mining methods that found equivalent or similar counterparts in the literature, we compared with the published results. We evaluated the effectiveness and scalability of our strategies as well. We also used our metrics to avoid the curse of dimensionality problem when aggregating high n-dimensional datasets. These metrics are discussed in Chapter 5.
- ***Dissemination.*** We disseminated the results of this research through submissions of papers to peer reviewed workshops and conferences [Alves & Belo 07a] [Alves & Belo 07b] [Alves et al. 06a] [Alves & Belo 06b] [Ferreira et al. 06] [Alves & Belo 05b] [Ferreira et al. 05]. The review process and discussions at these meetings were essential for further improvements of our MDM infrastructure.

1.5 Organization of the Dissertation

The rest of this dissertation is organized as follows:

- In ***Chapter 2***, we present data mining methods for mining intra and inter-dimensional patterns. We discuss the main differences between these two methods highlighting the basis for achieving MDM. The kernel of these methods relies on frequent pattern mining basis. We devised a database-like strategy for mining intra-dimensional patterns closer to RDBMS reviewing the major methods in the literature. To achieve a basic level for MDA, in this case, inter-dimensional ones, we extend the intra-dimensional approach with a hybrid strategy. Thus, we combined frequent patterns with sequential patterns for getting inter-dimensional ones. The basic aggregating function in these methods is counting ones. These patterns are motivated by the classical application of association rules on market basket analysis. Finally, we also ran several experimental studies to get an overall picture of these methods against different types of datasets. Preliminary versions of these studies were published in [Alves & Belo 05b] [Ferreira et al. 05].

- In ***Chapter 3***, we take a different approach for achieving MDM. We allowed more complex aggregating functions such as average and standard deviation. We restricted this strategy for handling single-dimensional mining, i.e., aggregation-based mining over one dimension at a time. All aggregations formed what we call profiles or signatures. We also reviewed signature-based concepts and applications. Furthermore, we started from signatures to extract anomalous patterns. To get such patterns we devised several mining methods over signatures. To differentiate normal from abnormal behavior we proposed a deviation-based strategy. To get abnormal behavior from groups or clusters, we proposed a clustering-based method. To complete this single-dimensional strategy we also have explored social or community behavior by exploring aggregating functions within a graph-based approach. These methods were developed through our working on two Telecommunication projects (FRATELO and SFonTEL). We evaluated the effectiveness and scalability of those methods on detecting potential fraud situations from a mobile company. These methods were published in [Alves et al. 06a] [Ferreira et al. 06].
- In ***Chapter 4***, we take some steps towards MDM, providing aggregation-based mining on n-dimensions. We provided a n-dimensional structure (cube) to allow multidimensional data analysis. We reviewed some classical data cubing methods, as well as, multidimensional association rules strategies in the literature. We devised a cube-based mining method which can compute an incremental cube, allowing concept hierarchy modeling, as well as, incremental mining of interesting relations by means of multidimensional and multilevel association rules. This method is also enhanced by a multidimensional measure of interestingness. As far as we are concerned, this method is the first one to support incremental cube-based mining over evolving databases. Finally, we ran several evaluation studies using real and synthetic datasets which demonstrated that this method is a promising OLAPing method. A preliminary version of this study was published in [Alves & Belo 07b].
- In ***Chapter 5***, we introduced two methods for advanced aggregation-based mining. We called them advanced methods in sense that both strategies push measures of interestingness during the cubing process over n-dimensional databases. The first

method was devised to evaluate correlation-mining in a multidimensional space. We review data cubing literature with respect to computing cuboids cells preserving semantics and compression levels. Both levels are essential requirements for multidimensional data analysis. Thus, the first method proposes a new notion for reducing cuboids aggregation by means of computing only the maximal-correlated cuboids cells, and presents the M3C-Cubing strategy that brings out those correlated cuboids. Our evaluation study shows that the method followed is a promising candidate for scalable data cubing, reducing the number of cuboids by at least an order of magnitude or more in comparison with that of closed ones. The second method introduced in this chapter evaluates interesting changes in high dimensional databases. We reviewed Gradient queries and added a new level of interestingness through exploring the most relevant (K) gradients in a multidimensional space. Thus, we have explored Top-K cuboid cells of interesting multidimensional gradients. There several studies on Top-K queries, but preference queries with multidimensional selection were introduced quite recently by [Dong et al. 06]. Furthermore, traditional Top-K methods work well in presence of convex functions (gradients are non-convex ones). We have revisited iceberg cubing for complex measures, since it is the basis for mining gradient cells. We also propose a gradient-based cubing strategy to evaluate interesting gradient regions in MDS. Thus, the main challenge is to find maximum gradient regions (MGRs) that maximize the task of mining Top-K gradient cells. Finally we presented a performance study indicating that our strategy is effective on finding the most interesting gradients in multidimensional space. The motivation for both advanced cubing methods comes from a broad range of applications, for instance business, stream, or scientific-data applications. These methods were published in [Alves & Belo 06b] [Alves & Belo 07b].

- In ***Chapter 6***, we conclude this dissertation with a brief summary of the work presented, discussed the main results achieved in this research, draws some conclusions and points future work directions as a continuation of this research.

Chapter 2

Mining Intra and Inter-dimensional Patterns

In this chapter, we cover two main strategies devised for mining intra and inter-dimensional patterns. Let us take as example a Retail Sales Warehouse [Kimball 02] to understand the applicability of those strategies. The fine grain of our sales warehouse is represented by a customer's retail sales ticket. Thus, the most granular data is an individual line item on a POS transaction. Consequently, we may think as strong candidate dimensions, the date, the product and the store. Further refinements in this model we head us to a retail sales schema as presented in Figure 2.

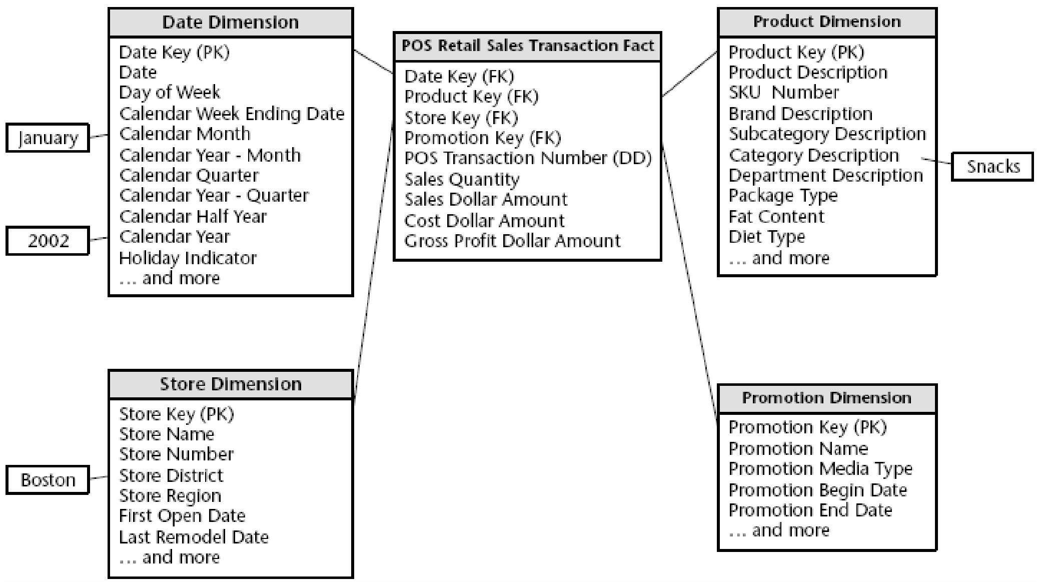


Figure 2. The retail sales star schema, example from [Kimball 02].

Let us say that the retail sales schema is done and now we are able to answer a couple of questions. With such schema we can answer queries concerning what was purchased at each store and under what conditions. However, ***this schema doesn't allow us to evaluate directly which products were sold in the same market basket together***. This evaluation is quite known by the data mining community research as *Market Basket Analysis* (MBA). In computational terms, MBA relies on devising effective and efficient *Frequent Itemset Mining* (FIMI) algorithms.

The retail sales fact table cannot be used easily to perform market basket analyses since SQL was never designed to constrain and group across line item fact rows [Kimball 02]. However, one can make use of some ***SQL-based FIMI strategy*** to overcome this difficulty [Alves & Belo 05a] [Alves & Belo 05b]. Another interesting issue is that usually MBA is high correlated with temporal patterns [Ale & Rossi 00]. Therefore, itemsets that seem to be quite frequent during the Christmas' day may not be frequent either in a day before, a day after, a week before or even in the same month (i.e. December).

From the above discussion we can summarize this chapter into two main statements concerning MBA from fact tables:

1. Once fact tables are usually stored on Relational Database Management Systems (RDBMS), ***how can one bring itemset mining into RDBMS server side, tanking control and management of data?*** To answer this question we propose a database-like strategy to mine intra-dimensional patterns.
2. Given that MBA is sensitive to time (or context) dimension, ***how can one bring time (or context) dimension into the FIMI process to smooth the bias added by temporal patterns***? To reveal time effects on MBA we devised a strategy based on mining inter-dimensional patterns.

In this chapter we present two mining strategies for answering the previous two questions. In section 2.1 we highlight FIMI from a SQL-like perspective. A new strategy based on the Pattern-growth mining [Han et al. 00] method is proposed in section 2.2, as well as, the complete procedural schema to make it available for running on RDMBS server side [Alves & Belo 05a]. Finally, in section 2.3, we devised a strategy which combines intra-dimensional patterns with sequence mining to obtain inter-dimensional patterns.

A preliminary version of this chapter appeared in two papers presented in [Alves & Belo 05b], and [Ferreira et al. 05] concerning to intra- and inter-dimensional patterns respectively.

2.1 Mining Fact Tables in RDBMS

The problem of finding all frequent itemsets [Agrawal et al. 93] given a Dataset D with a minimum support threshold S is the most time consuming task on association rule mining. In order to solve this problem, two ways are likely to be chosen:

1. One using algorithms that employed sophisticated in memory data structures, where the data is stored into and retrieved from flat files;
2. and another using algorithms that are based on SQL statements and extensions to query and update a database.

The former is very efficient when it is compared with the later. On the other hand, when we deal with data warehouse the size of tables is extremely huge for memory copy. Nevertheless, it becomes important for RDBMS to offer new analytic functionalities to support business intelligence applications.

There are a few implementations based on SQL [Rantzau 03] [Sarawagi et al. 98a] [Shang et al. 04] [Wang & Zaniolo 00] [Yoshizawa 00], but they have performance issues concentrated in two central points: candidate-set generation and test (Apriori-bottleneck); and table reconstruction of conditional pattern trees (FP-Growth-bottleneck).

In this section we do not intend to compare the effectiveness of itemset mining based on database programming with the memory ones. Instead, we purpose a solution for bringing the itemset mining process to the RDBMS server side, which generates the following contributions:

1. A ***procedural schema*** for itemset mining, so the process can be run as a batch script on the RDBMS server side.
2. A ***cascading approach*** working with single tables, and also avoiding the complexity of mining frequent itemsets from several multi-tables joins.
3. A ***pattern growth mining*** which doesn't surfer of several tables reconstruction (of conditional pattern trees).

2.1.1 Frequent Pattern Mining

The frequent pattern mining problem can be defined as follow: Given a set of items *I*, a transaction database *D* over *I*, and a minimal support threshold *S*, find all itemsets *F(D,S)*. Indeed, we are not only interested in the set of itemsets (F), but also in the actual supports of these itemsets. The most known implementation of frequent pattern mining algorithm is Apriori [Agrawal & Srikant 94]. Several Apriori-based algorithms have been purposed for getting better performance and I/O costs [Hidber 99] [Orlando et al. 01] [Orlando et al. 02]. Recently, an FP-tree based frequent pattern mining method, called FP-growth, developed by Han et al. [Han et al. 00], achieved high efficiency, when compared with the Apriori-like approaches. Basically, the FP-growth method adopts the divide-and-conquer strategy. It uses only two full I/O scans of the database, and avoids iterative candidate generation. In general terms, the mining process consists of making available the FP-tree data structure, and then FP-growth is applied over a FP-tree for getting frequent itemsets.

There are implementations that suggest enhancements into the frequent pattern mining in order to make the process interactive, constrained, and incremental [Cheung & Zaïane 03] [El-Hajj & Zaïane 03]. Those aspects will not be discussed in this chapter. Instead, we focus on the first issue, i.e., ***finding frequent itemsets closer to RDBMS***.

The above implementations cannot be applied directly on the main problem of this work, since they need to copy tables out from the database for proper execution. Besides, after its execution, the results must be load again to the database for getting suitable analysis.

2.1.2 Pattern Growth Mining

Pattern Growth Mining can be viewed as first mining frequent 1-itemset and then progressively growing each such itemset by mining its conditional pattern base, which implies first mining its frequent 1-itemset and then progressively growing each such itemset by mining its conditional pattern base, etc [Han et al. 00]. Thus, a frequent k-itemset mining problem can be transformed into a sequence of k frequent 1-itemset mining problems via a set of conditional pattern bases (see Figure 3). The main aspects of the algorithm can be summarized as follows:

1. For each node in the **FP-tree** construct its conditional pattern base, which is a sub-database constructed with the prefix sub-path set co-occurring with the suffix pattern in the FP-tree. FP-growth traverses nodes in the FP-tree from the least frequent item in *I*.

2. From each **conditional pattern base** construct its conditional FP-tree.
3. Finally, if the conditional FP-tree has a single path, simply enumerate all patterns. On the contrary run **pattern growth mining** recursively over the conditional FP-tree.

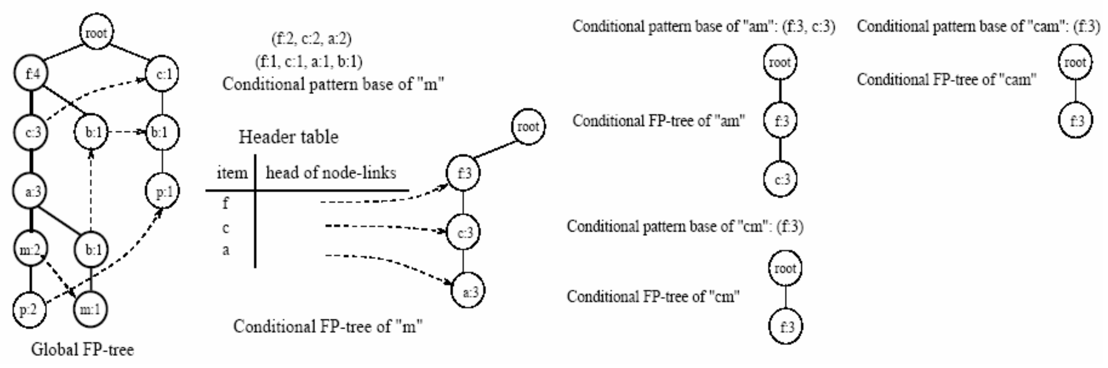

root
f:4
c:1
c:3
b:1
b:1
a:3
p:1
m:2
b:1
p:2
m:1
Global FP-tree
(f:2, c:2, a:2)
(f:1, c:1, a:1, b:1)
Conditional pattern base of "m"
Header table
item
head of node-links
f
c
a
root
f:3
c:3
a:3
Conditional FP-tree of "m"
Conditional pattern base of "am": (f:3, c:3)
Conditional FP-tree of "am"
Conditional pattern base of "cm": (f:3)
Conditional FP-tree of "cm"
Conditional pattern base of "cam": (f:3)
Conditional FP-tree of "cam"

Figure 3. The overall process of mining FP-trees, example from [Han et al. 00].

2.1.3 SQL-based Itemset Mining

There are a few SQL-based implementations that can be used to mine frequent patterns over large transactional tables [Agrawal & Shim 96] [Rantzau 03] [Sarawagi et al. 98a] [Wang & Zaniolo 00] [Yoshizawa 00]. Even so, all of them are based on nature of Apriori-like approach. There is another approach that uses FP-Growth [Shang et al. 04] in RDBMS. Nevertheless, the process of reconstructing conditional FP tables for large datasets may pose performance issues. Therefore, we must avoid the previous mentioned bottlenecks: candidate set generation and test; and table reconstruction. Moreover, we have designed a procedural schema for mining all patterns on the RDBMS server side. We examined those assumptions, and propose a new approach for pattern growth mining using database programming.

By using a pattern growth approach we are able to manage the first bottleneck. However, the current SQL implementation of FP-Growth [Shang et al. 04] cannot handle the second issue. Consequently, we need to provide a solution for pattern growth mining which must have the ability to: ***work in RDBMS server side, prevent multi-table joins and table reconstruction of conditional pattern trees.***

2.2 Extracting Intra-dimensional Patterns

From FIMI we can extract several kinds of patterns. The general one is the intra-transactional or intra-dimensional pattern. We use the same nomenclature as [Han & Kamber 06]. Thus, like in a DW, we refer to each distinct predicate in a rule as a dimension. For instance, an association rule likes: *Buys*(X, cod fish) => *Buys*(X, potato). This rule is an intra-dimensional pattern, since it contains a single distinct predicate (e.g., *Buys*) with multiple occurrences (i.e., the predicate occurs more than once within the rule). Such rules are commonly mined from transactional data in MBA.

2.2.1 Getting Frequent Itemsets from Large Transactional Tables

In order to provide itemset mining over large transactional tables on the RDBMS server side, we present a procedural schema by means of using several database programming such as stored procedures, SQL-cursors, and UDF functions. We also call this approach as a ***Pattern Growth mining with SQL-Extensions (PGS)***. The whole procedural schema can be found in [Alves & Belo 05a].

The whole process can be summarized into two main steps: one for generating the pattern tree (PT) and another one for mining all patterns from it PT. Table 1 shows the frequent 1-itemsets extracted from a transactional table (columns TID and Items). A new transactional table (column Freq. 1-itemsets), containing only records with frequent 1-itemsets, is created, and thus its related pattern tree too. Finally, Table 2 presents the pattern growth method applied over the pattern tree. For instance, giving that only items (column Item) are frequents, from its pattern tree, we work with only a subset (column SUBFP) for reaching its conditional pattern tree (column CONFP) and then enumerating all frequent patterns.

TID	Items	Freq. 1-itemsets
1	1, 3, 5, 6, 7	3, 5, 7, 1, 6
2	2, 3, 5, 6, 7	3, 5, 7, 6
3	1, 2, 3, 4, 5	3, 5, 1
4	3, 6, 7	3, 7, 6
5	1, 4, 5, 7	5, 7, 1

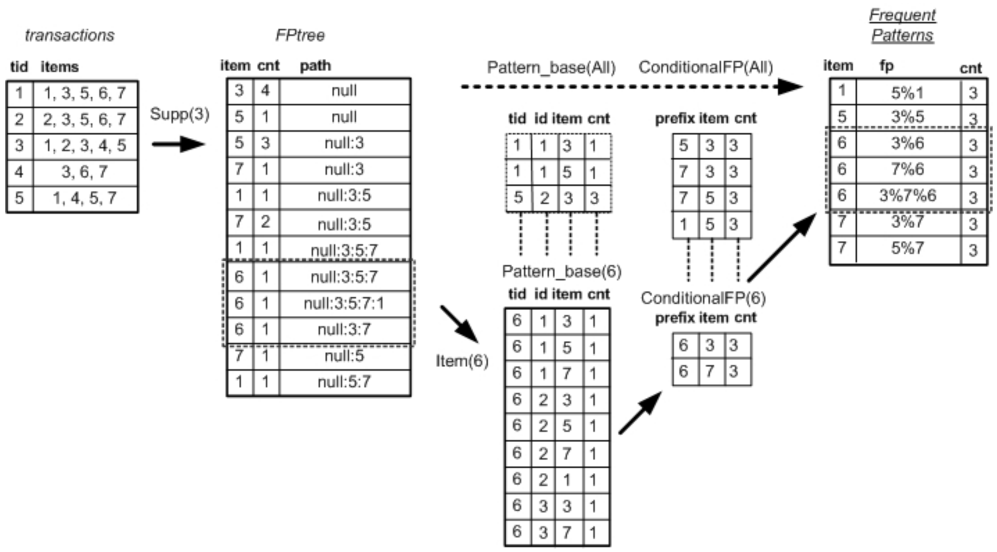
Table 1. A transactional database with a minimum support = 3.

Item	SUBFP	CONFP	PATTERNS
3	*Null*	*Null*	*Null*
5	R=1, R:3=3	3:3	3%5:3
7	R:3=1, R:3:5=2, R:5=1	3:3, 5:3	3%7:3, 5%7:3
1	R:3:5=1, R:3:5:7=1, R:5:7=1	5:3	5%1:3
6	R:3:5:7=1, R:3:5:7:1=1, R:3:7=1	3:3,7:3	3%6:3, 7%6:3, 3%7%6:3

Table 2. Extracting all patterns (R=root).

2.2.2 Generating Pattern-tree Table

Several tables are manipulated during the process of generating all itemsets (Figure 4 and Figure 5). They are built just once. In the recursive part, where pattern growth is applied, other structures are required, but they are created dynamically by using UDF functions and database cursors. In fact, they provide SUBFP (sub-path) tables for extracting single and not-single patterns.



transactions

tid	items
1	1, 3, 5, 6, 7
2	2, 3, 5, 6, 7
3	1, 2, 3, 4, 5
4	3, 6, 7
5	1, 4, 5, 7

FPtree

item	cnt	path
3	4	null
5	1	null
5	3	null:3
7	1	null:3
1	1	null:3:5
7	2	null:3:5
1	1	null:3:5:7
6	1	null:3:5:7
6	1	null:3:5:7:1
6	1	null:3:7
7	1	null:5
1	1	null:5:7

Pattern_base(All)

tid	id	item	cnt
1	1	3	1
1	1	5	1
5	2	3	3

ConditionalFP(All)

prefix	item	cnt
5	3	3
7	3	3
7	5	3
1	5	3

Pattern_base(6)

tid	id	item	cnt
6	1	3	1
6	1	5	1
6	1	7	1
6	2	3	1
6	2	5	1
6	2	7	1
6	2	1	1
6	3	3	1
6	3	7	1

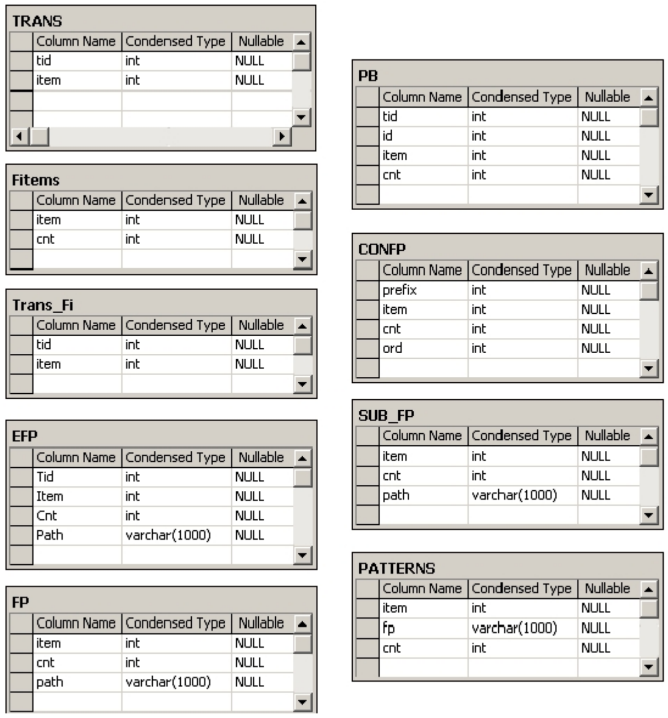
ConditionalFP(6)

prefix	item	cnt
6	3	3
6	7	3

Frequent Patterns

item	fp	cnt
1	5%1	3
5	3%5	3
6	3%6	3
6	7%6	3
6	3%7%6	3
7	3%7	3
7	5%7	3

Figure 4. An overall picture of the tables involved in the PGS strategy.



TRANS

Column Name	Condensed Type	Nullable
tid	int	NULL
item	int	NULL

Fitems

Column Name	Condensed Type	Nullable
item	int	NULL
cnt	int	NULL

Trans_Fi

Column Name	Condensed Type	Nullable
tid	int	NULL
item	int	NULL

EFP

Column Name	Condensed Type	Nullable
Tid	int	NULL
Item	int	NULL
Cnt	int	NULL
Path	varchar(1000)	NULL

FP

Column Name	Condensed Type	Nullable
item	int	NULL
cnt	int	NULL
path	varchar(1000)	NULL

PB

Column Name	Condensed Type	Nullable
tid	int	NULL
id	int	NULL
item	int	NULL
cnt	int	NULL

CONFP

Column Name	Condensed Type	Nullable
prefix	int	NULL
item	int	NULL
cnt	int	NULL
ord	int	NULL

SUB_FP

Column Name	Condensed Type	Nullable
item	int	NULL
cnt	int	NULL
path	varchar(1000)	NULL

PATTERNS

Column Name	Condensed Type	Nullable
item	int	NULL
fp	varchar(1000)	NULL
cnt	int	NULL

Figure 5. The PGS database scheme.

We mean single patterns those which are enumerated directly from its **conditional FP table** (CONFP) (for instance 3%5), meaning a co-occurrence of item 3 with item 5, without handling sub-path tables. A pattern such as 3%7%6 is extracted by combining those items that co-occurs added with it is respective sub-path tables [Han et al. 00].

As an pattern growth approach the first step requires a **pattern-tree** structure also called FP-tree. Even though FP-tree is a compact structure, it is unlikely to build such structure in memory for large databases. Consequently, using RDBMS capabilities like buffer management, query processor or SQL-Extensions, it is possible to take advantage of those mechanisms avoiding size considerations of data, in this particular case, FP (pattern-tree) tables.

The construction of **FP table** is set up on the following steps:

1. Based on a given support threshold (s), frequent 1-itemsets are selected from the transactional table TRANS.
2. A new transaction table TRANSFI is created based on transactions which contains those frequent 1-itemsets.

3. From the TRANSFI table, an EFP table which stands for Extended FP is built as a preprocessing step for reaching an FP table.
4. Finally, the FP table is created by means of an SQL expression, with proper aggregate function over EFP table.

The EFP table is an interesting approach for getting FP table, since it avoids for each frequent item to be tested if it should be or not inserted into FP table [Shang et al. 04].

```
# A piece of the Pattern TREE (FP) source code #
PROCEDURE EFP
DO with (EXISTS TRANSFI)
CREATE TABLE EFP (item, cnt, path)
CREATE TABLE FP (item, cnt, path)
DECLARE
  BEGIN
   count =  1
   curpath = null
   c transfi CURSOR for TRANSFI
   FOR each row in c transfi
   BEGIN
     curpath = curpath + ':' + c transfi.item
     INSERT INTO EFP
     values(c transfi.item, count, curpath)
   END
   SELECT item, sum(cnt) as cnt, path
   INTO FP
   FROM EFP
   GROUP BY item, path
  END
```

Algorithm 1. Generating Pattern-tree tables.

2.2.3 Mining Pattern-tree Tables

For mining FP table it is necessary to build two more auxiliary tables which are the pattern base (PB) and conditional FP table (CONFP). We present an approach where CONFP table is built based on simple SQL with proper aggregate functions over PB table. On the other hand, SQL-based FP-Growth [Shang et al. 04] demands several reconstruction processes for those tables. It is almost impractical to create those tables several times. Therefore, we use an approach for getting sub-paths by means of UDF functions and database cursors with its respective support threshold over the SUBFP table (a SUBset of FP table).

SUBFP is a table that contains only rows from FP table which have itemsets enclosed in CONFP. By doing so, we can reduce the search space for getting sub-paths directly from all items in FP table, and also, avoid several reconstruction of PB, FP and CONFP tables. The size of FP is reduced significantly by using SUBFP mainly when dealing with low support thresholds on large datasets. The following steps are required for mining pattern tree table [Alves & Belo 05b]:

1. Taking as input the same support threshold defined in 2.2.1, creates the related tables PB, CONFP and SUBFP.
2. Update the column (pos) in CONF which keeps the position of each item. This is useful for getting sub-path databases in such way that it preserves the order of the items on SUBFP table. This is important for using UDF (table-valued functions).
3. Extract single patterns by enumerating the prefix-item stored on CONFP table. This also creates the PATTERNS table.
4. Extract not-single patterns by applying pattern growth over CONFP.
5. In **Fragment Growth** step, each prefix-item is extracted from SUBFP table and verified by two UDF functions. One for generating the **sub-path databases** (function getTable_pb) and other for getting the **node support** associated to each prefix-item sub-path (function getNodeSupp). Those functions coupled with the SUBFP table play an important role for extracting all frequent patterns, and also avoid the re-construction of PB, FP and CONFP table for each prefix-item sub-path.

```
# A piece of the Fragment Growth source code #
   DECLARE pg subPath CURSOR for
     SELECT *
     FROM getTable_pb(@v_prefix,@v_item) order by ord
     SELECT list pg item  = pg subPath.item
     FOR each row in pg subPath
     BEGIN
   SELECT node path = pg subPath.item+’%’+ c confp.item
       SELECT node supp =
        getNodeSupp(pg subPath.prefix, node path)
       SELECT pat item = pg subPath.item
       SELECT pat fp = node path+'%'+ pg subPath.prefix
       SELECT pat cnt =  node supp
       SELECT exist_pat = (
                   SELECT count(*)  FROM PATTERNS
                   WHERE item=pat item and fp=pat fp)
         INSERT INTO PATTERNS (item,fp,cnt)
         VALUES (pat_item, pat_fp, pat_cnt)
       SELECT list pg item = list pg item
                +’%’+ pg subPath.item
     END

UDF FUNCTION getNodeSupp  (@item, @path)
```

```
RETURNS @node_supp ## -> node support
BEGIN
  DECLARE @supp int
  SELECT @supp = (SELECT sum(cnt)
              FROM   SUB FP
              WHERE item=@item and
              path LIKE '%'+@path+'%')
  RETURN(@supp)
END

UDF FUNCTION getTable pb (@prefix, @item)
RETURNS TABLE ## -> sub-path databases
AS   RETURN
       SELECT prefix,item,cnt,ord  FROM   CONFP
       WHERE  prefix=@prefix and
        item<>@item ##-> criteria for sub-path mining
```

Algorithm 2. Mining Pattern-tree tables.

2.2.4 Evaluation Study

In order to evaluate PGS we compare our results with an Apriori (K-way join) and improved SQL-based FP-growth (EFP). Those algorithms were chosen in sense that they present the basis on the most known itemset mining implementations based on SQL [Agrawal & Shim 96] [Rantzau 03] [Sarawagi et al. 98a] [Shang et al. 04] [Wang & Zaniolo 00] [Yoshizawa 00]. Both algorithms were implemented to the best of our knowledge based on the published reports on the same machine and compared in the same running environment. The former implementation uses a candidate k-table Ck, which is a slow process for generating all joins and tables. So, when dealing with long patterns and large datasets the K-way join seems to be not efficient. The EFP avoids candidate-set generation been more competitive in low support scenario. However, it demands several tables' reconstruction.

Our PGS strategy takes the other way around, beyond pure SQL to SQL-Extensions. Also getting sub-paths databases, and restricting the search space for finding frequent itemsets by means of using an SUBFP table coupled with UDF functions. Consequently, we don't need to materialize PB, FP and CONFP tables several times.

Datasets

We use the synthetic transaction data generation described in [Agrawal & Srikant 94] for generating transactional tables. The nomenclature of these data sets is of the form TxxIyyDzzzK. Where xx denotes the average number of items present per transaction, yy denotes the average support of each item in the data set, and zzzK the total number of transactions in K (1000's). Table 3 summarizes those datasets.

Datasets	Dist. Items	Nof. Rows	Avg.1-it.sup	Max.1-it.sup
T5I51K (1)	775	5.112	6	41
T5I5D10K (2)	873	49.257	56	399
T25I10D10K (3)	947	245.933	259	1468
T25I20D100K (4)	981	2.478,55	2526	13.917

Table 3. Transactional datasets generated.

Performance Comparison

We describe our approach PGS, comparing it with K-Way-join and EFP. Our experiments were performed with Microsoft SQL Server 2000 (v.8.0). The machine was a mobile AMD Athlon ™ 2000+ 645MHZ, 224 MB RAM. The performance measure was the execution time 'the logarithm of the execution time (log(milliseconds))' of the algorithm applied over the four datasets with different support thresholds. We took that log scale in order to get a better view of the performance comparison among all approaches, since PGS has good response time. Figure 6 shows the total time taken by the all approaches.

Based on the performance information provided in Figure 6 we can make the following observation: ***PGS can get competitive performance out of EFP and K-way-join***. K-way-join has low performance when dealing with large datasets. Besides, when the support decrease the length of frequent itemsets increase causing expensive joins with the transactional tables. Therefore, the other two approaches perform better than K-way-join.

The results of PGS and EFP answered our second issue. The former doesn't use table reconstruction, getting good response time. On the other hand, the latter suffers considerably by working with several table materialization reconstructions. By those preliminary results we have accomplished our main goals with the proposed PGS strategy.

The store procedures which deal with the construction of tables FP and CONFP are the most time-consuming tasks. They respectively took, for each dataset, 35%, 50%, 75% and 85% of the total execution time. Nevertheless, the time for the whole process was quite competitive and those tables are built only once. In order to speed up even more the whole process, we also have been applied two clustered indexes on tables CONFP and SUBFP.

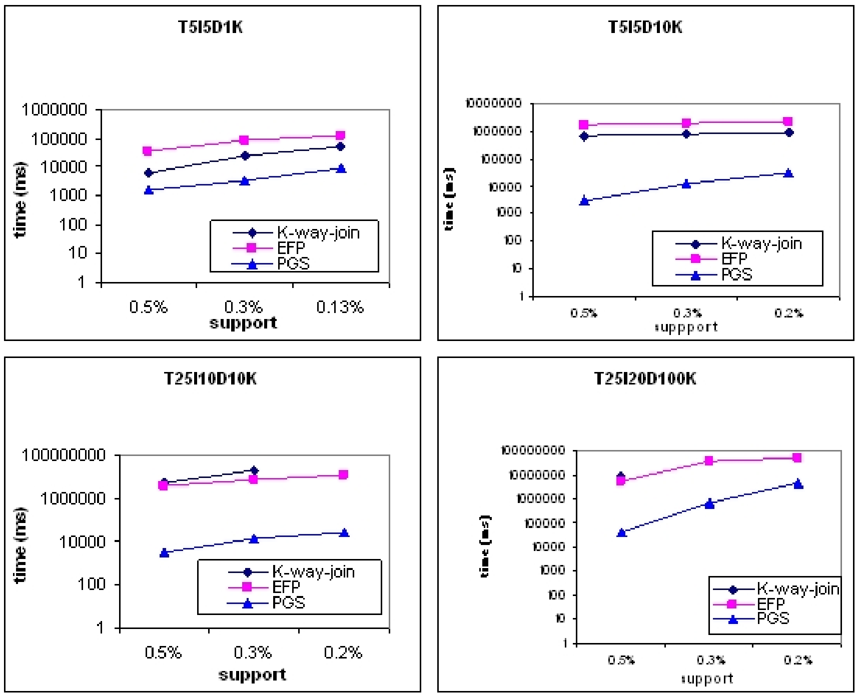


Figure 6. A comparative study. PGS runs competitively in sparse and dense datasets.

Foodmart Warehouse

FoodMart Warehouse is a sample database provided by Microsoft SQL Server 2000. One can use Analysis Services for applying OLAP and Data Mining techniques over data warehouses. However, only Clustering and Decision Tree methods were available in this DB version. Just quite recently was introduced an Apriori-implementation for MBA in SQL 2005 within the Business Intelligence Development Studio (http://www.sqlserverdatamining.com/DMCommunity/).

We applied PGS for mining frequent patterns over FoodMart database. It works only in first step of association rules, which means generating all frequent itemsets. For getting all the rules, one can program a store procedure using the pseudo-code in [Agrawal et al. 93]. We choose this last example for presenting some results with a real application. Although we know that its size is smaller than the largest one showed in a previous section, we also can reach interesting itemsets. Therefore we omit the performance comparison among all approaches.

It was used the fact table (sales_fact_1998) as the transactional table. This table has 164.558 tuples with five dimensions (product, time, customer, promotion and store). Before using PGS, we must define which dimensions in the fact table will be used as the transaction identifier (tid) and the set of items. Thus, the *tid* was set to the customer dimension and items to the product dimension. Furthermore, there are 1.559 distinct products distributed along 7.824 customers. The most frequent product was 277 Great English Muffins (143) and the less one was 1559 CDR Apple Preserves (43). It was executed several supports from (0.05%) to (0.01%). The most interesting itemset was 282%232 means Best Choice Salsa Dip and Great Wheat Bread in low level hierarchy. Given that the fact table was so sparse, the itemsets was selected only with very low support.

2.2.5 Final Discussion

In the previous section, it was presented a pattern growth mining implementation which takes advantage of SQL-Extensions for getting intra-dimensional patterns. Most of the commercial RDBMS vendors have implemented some features for SQL-Extensions. Integrating data mining in RDBMS is a quite promising area. Frequent pattern mining is the basic task in data mining. There are several memory-approaches to mine all patterns. However, some few efforts have been made on database perspective, in those cases, only pure-SQL.

Given the large size of database like data warehouse, it is interesting to take advantage of databases features in order to manage and analyze large tables. We worked in this direction providing an approach with SQL-Extensions. By doing so, ***we can achieve competitive results and also avoid classical bottlenecks: candidate set generation and test (several expensive joins), and table reconstruction***. The Store Procedures that deals with the construction of the tables, FP and CONFP, are the most time-consuming tasks, taking 35%, 50%, 75% and 85% respectively for each synthetic dataset (1, 2, 3, 4) from the small one to the large one. Nevertheless, the time for the whole process was quite competitive. Moreover, it was used the FoodMart Warehouse with several supports in order to illustrate interesting intra-dimensional patterns.

One issue that is controversial is the code portability, in sense that PGS is tightly dependent of the database programming language with SQL-Extensions. On the other hand for huge databases, one might be interested to take all the advantages offered by the RDBMS.

As future research, one should work on method enhancements for making this strategy more interactive, constrained and incremental. Furthermore, some work on improving the whole performance of PGS by using Table Variables (TV), which allows mimicking an array, instead of using database cursors, it is also very interesting topic for further research with PGS.

2.3 From Intra-dimensional to Inter-dimensional Patterns

2.3.1 Motivation

In data warehouses, data records (in fact tables) are typically associated to one or more contexts or dimensions. These records can be related to distinct dimensions (or predicates) such as items, time and location. Classical examples are customer transactions in supermarkets, logs of network, clickstreams, or events related to manufacturing in industry. When analyzing such events within its context (i.e. associated dimensions), they provide valuables sources to explore, not only by its frequency and the co-occurrence, but further, inter-dimensional patterns that arise along each context associated to it. However this kind of inter-dimensional patterns can not be induced directly by using the method discussed in section 2.2. Context information is usually discarded by intra-dimensional mining methods [Agrawal & Srikant 94]

A typical example of this type of pattern is that one obtained from stock market database (Example 1). if company A goes up (day-0), company B goes down (day-1) then with probability of P% company C will go down (day-4). Although, this type of pattern provides more accurate time information than rules that only express a temporal order (e.g. A, B and C will happen within 5 days) or sequences (e.g. B happens after A and C after B) , such accuracy may also incurs some bias into the mining process. Let's take the Example 1, where the company B goes down sometimes at day-1 ($A_0 \rightarrow B_1 \rightarrow C_4$ with probability of $\frac{P}{2}$%) and other times at day-2 ($A_0 \rightarrow B_2 \rightarrow C_4$ with probability of $\frac{P}{2}$%). Even though the pattern trend is still the same, this may result into a not frequent pattern (i.e. both patterns do not achieve the minimum required probability P% when both of them are evaluated in separated manner). To overcome this situation we propose to mine inter-dimensional patterns in the form: if company A goes up in one day, company B goes down on a subsequent day, then with probability of P% company C goes down after B and the mean distance between A and C is μ and its respective standard deviation is σ. Furthermore, we also can obtain the mean distance of all item of a pattern. ***Our mining strategy relies in the combination of frequent pattern and sequence pattern mining to extract inter-dimensional patterns***.

2.3.2 Related Background

The task of association rule mining is well studied problem with many proposed algorithms [Agrawal & Srikant 94] [Han et al. 00] [Pei et al. 00] [Zaki & Hsiao 02]. Other studies are derivative from this original problem, like multilevel, with constraints or quantitative rules. Sequential pattern mining refers to the task of finding sequence patterns. The first algorithm to tackle this problem was presented in [Agrawal & Srikant 95] and other algorithms [Srikant & Agrawal 96] [Zaki 01] [Garofalakis et al. 99] [Pei et al. 01] [Ayres et al. 02] were successively proposed to improve this task. The transactions are aggregated by customer and ordered in time. Each customer is described by the respective sequence of transactions.

Frequent patterns are extracted having into account the respective order of their items along the sequence. Viewing each sequence as a transaction, the extracted patterns are intra-transactional (or intra-dimensional) in their nature. Other algorithms that incorporate temporal information into association rule mining have also been proposed. In [Ale & Rossi 00] it was proposed an algorithm that only considers the items' lifetime for supporting counting purposes, since an item may not be frequent in the overall time but be frequent in its life time window. The notion of cyclic association rule is presented in [Özden et al. 98]. Rules like *coffee and donuts are frequent from 7AM to 10 AM"*, express a regular variation (hourly, daily, weekly…) and can be considered as a cycle. None of the previous algorithms are inter-transactional (or dimensional).

Inter-transactional patterns were introduced in [Lu et al. 00]. In this work, the notion of intra-transactional rules was extended to the multidimensional space. An Apriori [Srikant & Agrawal 96] based algorithm, called EH-Apriori was proposed to mine such rules. The algorithm works by moving a sliding window across the database of transactions. Each item is mapped into an integer identifier (extended item), being tagged by the offset of the respective original transaction within the window. All the integer identifiers are projected into new transactions (mega-transactions) forming a transformed transactional database. This database is then mined for the extraction of frequent itemsets and its respective association rules. In [Tung et al. 03] a new algorithm called

FITI (``First Intra Then Inter'') was presented, showing a better performance than EH-Apriori. The FITI algorithm works in three phases:

1. First, it starts by mining and storing frequent intra-transactional itemsets.
2. In a second phase, the stored itemsets are mapped into integer identifiers and the database is transformed using these identifiers. Once again, a sliding window is used to restrict the span which rules may have.
3. Finally, the transformed database is mined for inter-transactional itemsets, where some constraints are imposed to the association rules, in order to limit the number of interesting rules.

In [Berberidis et al. 04] an approach based in [Lu et al. 00] is used to predict rare items. Here, the user selects an item that wants to predict. Antecedent periods of the selected item are used to generate mega-transactions that will compose the transformed database.

The above described algorithms for inter-transactional (or inter-dimensional) mining make use of an item or itemset mapping to force the order among the itemsets. In our proposed strategy, association mining is used to extract interesting itemsets present in a transaction. Applying a sliding window, those itemsets are aligned, and compose a set of sequences, which are then mined in order to find frequent sequence patterns. Finally sequential rules, that express frequent actions occurring within a dimensional window, are obtained.

2.3.3 Inter-dimensional Patterns

In this section we provide definitions which will allows us to express inter-dimensional patterns among different transaction records, followed by the problem of mining such pattern.

Definition 1 (**n-dimensional database**) Let Ω be a set of events, generally called items. Let D the dimensional attribute and Dom(D) the domain of D. DB is a n-dimensional database and contains instances in the form (d, e), where d $\in$ Dom(D) and $e \subseteq \Omega$. As an example of dimensional attributes we have time, distance and temperature. It is assumed that Dom(D) is R or N, that $d_{i+1} > d_i$ and $(d_{i+1}-d_i)$ is equal for all i.

Definition 2 (**frequent itemset**) I is a frequent itemset, where I = $\{i_j,...,i_k\} \subseteq \Omega$, if it occurs in a number of transactions θ(minimum support). This is also the same notion of an intra-dimensional pattern discussed in section 2.2.

Definition 3 (**sequence**) A sequence s = $s_1 * s_2 * ... * s_k$ is an ordered list of items, where $s_i \subseteq \Psi$ (an alphabet of items), i ∈ {1,…,k}. * is a wildcard that corresponds to zero or more arbitrary items from Ψ. The length of s, |s|, is the number of non wildcard in s. As an example of a sequence we have abc**e*f, which has a length of 5. The items are contained in Ψ. In the next section we will see that $\Psi \neq \Omega$.

Definition 4 (**sequence database**) A database of sequences P is a set of instance tuples in the form (s_{id}, s), where s_{id} is the sequence identifier and s the respective sequence. |P| is the size (number of sequences) of P.

We may look to the sequence identifier as a customer or time identifier. S follows definition 3 of a sequence. Furthermore, a sequence is frequent if it occurs in a number of sequences greater than a **sequence minimum support** φ. The **cover list** of an itemset or sequence pattern is, respectively, the set of the transactions and sequence where it occurs.

A **sliding window** w, is a dimensional frame that is used to limit the span of the inter-dimensional patterns, i.e. the range of interaction between transactions. This window is imposed for performance issues, since it limits the search space. That is, rules that span too much and have no significant interest.

The problem of mining inter-dimensional patterns can be formulated as follow: Given a database DB, an itemset minimum support θ, a window size w, a sequence minimum support φ, a minimal rule confidence c (or other measure of interestingness), find all sequence rules (inter-dimensional patterns) in the form $(A_1 \rightarrow ... \rightarrow A_n) \Rightarrow (C_1 \rightarrow ... \rightarrow C_m)\{\mu, \sigma\}$. $A_i, C_j (1 \leq i \leq n, 1 \leq j \leq m)$ are in the form $i_1 * ... * i_p$, where $i_k \subseteq \Omega (1 \leq k \leq p)$. μ is the mean distance between A_1 and C_m. σ is the standard deviation of the distance.

2.3.4 Mining Inter-dimensional Patterns

Our method is divided into three steps, enumerated as follows:

1. Database transformation,
2. Sequence conversion and mining, and
3. Sequence rule extraction

To get a better understanding of the proposed strategy, for next sections we take time as the dimensional attribute.

Database transformation

The main goal in this step is to extract the relevant information in the context of the problem described in section 2.3.3. This transformation corresponds to the selection of relevant items for each transaction. This can be achieved through applying a function f, where f(DB)=DB'; and DB and DB' as the original and the transformed databases respectively. We choose a generic function f, suitable to be applied in many domains and in particular to the time domain. This function extracts all frequent intra-transactional patterns (greater than a pre-defined size) present in a transaction for a minimum support threshold θ. An example of an alternative function is the, **correlation-function**, where only sets of items that have a pair-wise correlation greater than a particular threshold are evaluated. The motivation to use a function f, is that by enumerating all frequent intra-dimensional patterns in a transaction, all the possible pattern combinations are tested out. Table 4 and Table 5 illustrate the utilization of such function.

d	t
...	...
d_i	<1 2 3>
d_{i+1}	<4 5 6>
d_{i+2}	<7 8 9>
...	...
d_j	<1 2 6>
d_{j+1}	<4 5 10>
d_{j+2}	<8 9 15>
...	...

Table 4. Subset of an original database DB.

d	t
…	…
d_i	<1 2> <1 3> <2 3> <1 2 3>
d_{i+1}	<4 5> <4 6> <5 6>
d_{i+2}	<8 9>
…	…
d_j	<1 2> <1 6>
d_{j+1}	<4 5> <5 10> <4 10> <4 5 10>
d_{j+2}	<8 9>
…	…

Table 5. Subset of a transformed database DB'.

After obtaining all frequent patterns (intra-dimensional ones) greater than size one, we are able to enumerate all inter-dimensional patterns. Let's consider that transactions (d_i, d_{i+1}, d_{i+2}) are within window W_k, and transactions (d_j, d_{j+1}, d_{j+2}) within W_p, ($K \neq p$), then <1 2> → <4 5> → <8 9> is an inter-dimensional pattern (Table 5).

The method explained in section 2.2 is applied here to enumerate all intra-dimensional patterns [Alves & Belo 05a] [Alves & Belo 05b]. Next, for each transaction identifier, the cover list of all patterns in the set of all intra-dimensional patterns is scanned. When the pattern appears in a given transaction, it is reported out for that particular transaction.

Sequence Conversion and Mining

First, the database DB' is converted into a database of sequences P. Then, P is evaluated in order to obtain sequence patterns (inter-dimensional ones).

The process starts by mapping each itemset to an integer identifier. The hash table (Table 6) containing all itemsets (alphabet Ω) | identifier (alphabet Ψ) correspondence is kept during this evaluation.

itemset	<1 2>	<1 3>	<2 3>	<1 2 3>	…	<4 5>	<4 6>	<5 6>	<8 9>	…
Id	1	2	3	4	…	8	9	13	14	…

Table 6. Itemset mapping hash table.

Next, database DB' is scanned and for each w, consecutive transactions with its correspondent itemsets identifiers are projected into a dimensional sequence. These sequences will compose the database P. The obtained sequences are non-overlapped, i.e. mutually exclusive, which avoid the bias introduced by the overlapping regions of the sequence. Therefore, if a sequence starts at d_p the sequence will start at d_{p+k}, which results in the following relation |DB'|≥ w x |P|. Taking the example given in Table 4 and Table 5, for the window w_k, we will have the sequence identifiers: 1 2 3 4 8 9 13 14.

Finally, we are able to use any sequence mining algorithm to mine all the sequence patterns that are frequent for a given sequence support φ. In this proposal we make use of gIL [Ferreira et al. 05]. This algorithm makes use of a **sequence transitivity extension** property. Thus, in order to get a pattern A → B → C, the pattern A → B needs to be combined with the pattern B → C. This allows us to keep a track of the mean distance among all the items.

Rule Extraction

In the rule extraction step, sequence patterns are filtered out and rules are generated. Only rules that fulfill constraint of interestingness are reported. Pattern distance can be problematic in where standard deviation is high. Thus we devised a new filter **coefficient of variation of distances**, $cvd = \frac{\sigma}{\mu}$, applying a threshold min_cvd to remove sequences which express high distance variation. In addition, we also need to smooth redundant patterns. Redundant patterns are itemsets which are contained in subsequent itemsets originated in the same transaction. For instance, the patterns represented by sequence identifiers {1, 2, 3} in Table 6 are contained in sequence {4}, and all of them belong to the same transaction.

Following the definitions presented in [Spiliopoulou 99], a sequence rule is obtained by splitting a sequence pattern s in two adjacent parts: (l,r), where s=l.r (. is the concatenation of two sequences) . The obtained rule has the form: l → r. While testing a sequence s for different rules, the split into l and r parts, should be done using different combinations of the subsequences of s. However, the order of those subsequences should be preserved. As an example, for the sequence abcd, the following combinations are valid ones: abc → d, ab → cd, and a → bcd. The typical measures of interests used in association mining, can also be applied in sequence mining. For a database of sequences P, we may have:

1. support sup (l → r) = $\frac{|l.r|}{P}$

2. confidence conf (l → r) = $\frac{\sup(l.r)}{\sup(l)}$

3. lift = $\frac{conf(l \to r)}{\sup(r)}$

2.3.5 Evaluation Study

To evaluate our method we will use several transactional datasets where the dimensional attribute is time. The datasets were generated according to the IBM data generator [Agrawal & Srikant 94], which has been widely used for the evaluation of frequent pattern mining algorithms (association & sequence). The datasets characteristics are illustrated in Table 7 and Table 8. The algorithms were implemented in database-like (T-SQL) for mining frequent patterns [Alves et al. 05b] and in C++ for getting sequence patterns [Ferreira et al. 05]. All experiments were performed on 1.5Ghz Intel Centrino machine with 512Mb (main memory), within Windows XP environment.

D	Number of transactions
T	Average size of transactions
I	Average size of max potentially large itemsets
L	Number of max potentially large itemsets
N	Number of items

Table 7. Parameters used by the dataset generator.

Dataset	D	T	I	L	N
DS1K	1K	10	5	0.02K	100
DS10K	10K	5	5	0.1K	70
DS50K	50K	5	5	0.5k	500
DS100K	100K	5	5	1k	500

Table 8. Datasets generated for the evaluation study.

Table 9 and Table 10 show the experimental values obtained in different steps of the method. Table 9 shows that for each dataset the value of the itemset support threshold (θ) and the respective running time to get all frequent patterns (intra-dimensional ones). It is also presented the values used and obtained while applying the sequence conversion step. The Max.Item refers to the number of different itemsets (with size greater than 2), W is the window size, Nº.Ws the number of windows (with size greater than one). The last column shows the windows average length.

Dataset	θ%	Time(s)	Max.Item	W	Nº.Ws	Avg.W.Len
DS1K	10	1.6	44	2	500	13.9
DS10K	10	58.3	245	4	2500	20.4
DS10K	50	205	75	3	2721	4.3
DS50K	0.5	1604	171	3	16513	27
DS100K	1	50614	56	2	46188	8

Table 9. Experimental values for the itemset mining and sequence conversion steps.

Table 10 presents the results while applying the sequence mining step. For each support threshold (φ), it is reported the computational time used to obtain frequent sequences along with their characteristics (quantity and average size). The latter column is obtained after removing redundant itemsets.

Dataset	φ%	Time(s)	Nº Sequences	Avg.Size
DS1K	2	4.6	32729	4.6
DS1K	5	3.2	2520	3.2
DS1K	10	2.8	460	2.8
DS10K	0.4	46	45360	3.9
DS10K	0.5	0.27	1388	3.0
DS10K	1	0.14	621	2.9
DS50K	5	27.6	8049	3.5
DS100K	1	78.5	6422	3.6

Table 10. Experimental values for the sequence mining step.

Finally, in Figure 7, we show the distribution plot of the cvd measure for all patterns in the DS50K dataset. We can see that from a total of 8049 sequence patterns, 1039 have a cvd greater or equal than 1; 5398 less or equal than 0.85; 1556 less or equal than 0.5; and only 3 have a cvd bellow 0.3. Figure 8 presents the distribution plot of the confidence and lift measures for the set of sequence rules extracted from the DS50K dataset.

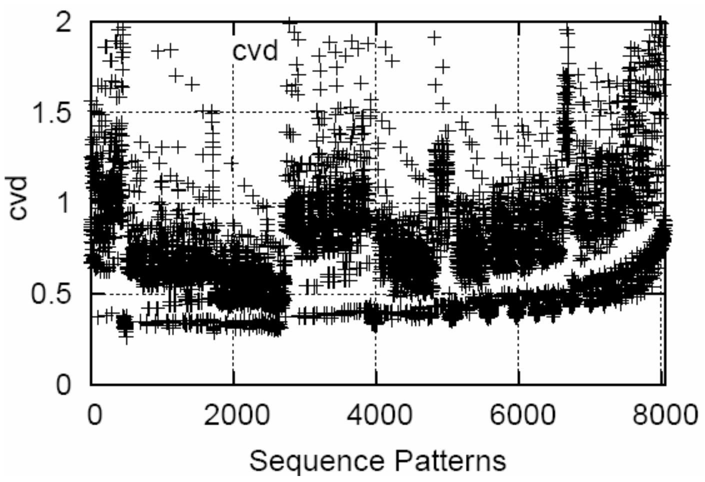


Figure 7. Distribution plot of the cvd measure for the DS50K dataset.

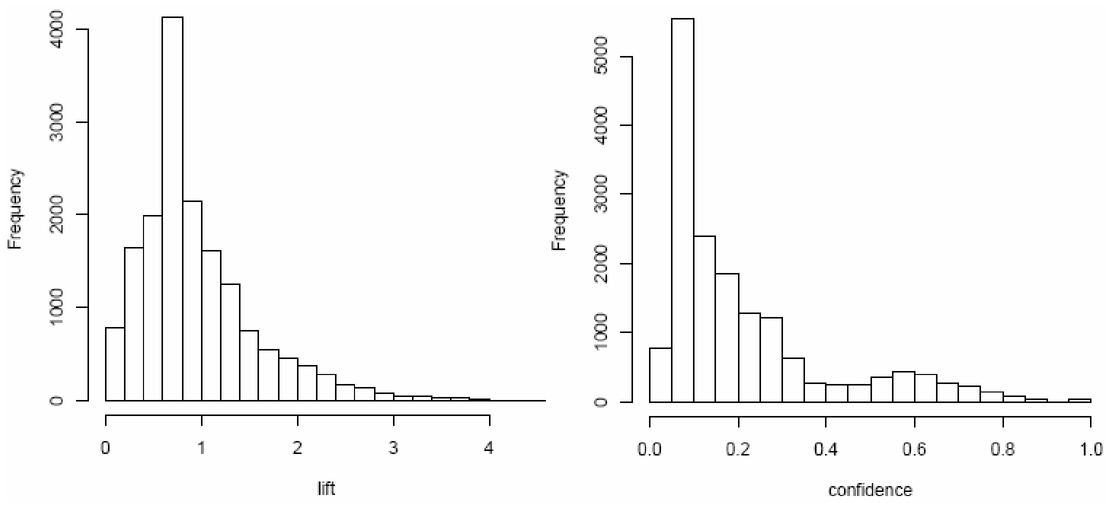


Figure 8. Distribution plot of the lift and confidence measures for the DS50K dataset.

2.3.6 Final Discussion

Previously we have introduced a strategy to extract inter-dimensional patterns from dimensional databases. This strategy combines two well known data mining tasks: frequent and sequence mining. First, a filter function was applied to the transactional database in order to obtain the relevant information (intra-dimensional patterns) from each transaction. This function extracts all frequent itemsets contained in a given transaction. Nonetheless, depending on the problem itself, different functions can be applied. Then, based in the extracted itemsets a database of sequence is built. Each sequence corresponds to a dimensional frame (particularly in this discussion we considered time as our dimension of interest). Finally, sequence rules fulfilling measures of interestingness are reported out. Those sequence patterns (inter-dimensional patterns) are augmented with two measures, μ and σ. The first measure corresponds to the mean distance between the first and last itemsets of the pattern and the later one its respective standard deviation of the distance. Optionally, the mean distance can also be reported. When compared with the offset based pattern description [Lu et al. 00] [Tung et al. 03], μ and σ, provides more flexibility and accuracy description with respect to the dimensional behavior of those patterns.

The introduction of a measure like cvd allow us to reject patterns that presents significant distance variation, and thus, avoiding the representation of misleading patterns. The performance of the strategy proposed in this section is tightly related to the algorithms used, the database characteristics and the parameters used on each step. The main advantage of this approach is its simplicity, where neither novel algorithm nor complex itemset labeling features are required.

So far, the cvd filter had been used as post-processing step of the sequence mining step. Although the performance of this strategy does not imply a bottleneck figure, it can be further improved. An interesting issued for future study is to push directly this pruning mechanism in the mining process. This is non trivial task, since this measure is neither monotonic nor anti-monotonic. Ideally, one should be concerned to devise a unique support measure, i.e. a measures combining both frequent and sequence threshold for mining inter-dimensional patterns.

2.4 Summary

In this chapter we covered two mining strategies devised for extracting intra and inter-dimensional patterns. The kernel of such strategies is based on FIMI theories.

Most of the itemset mining approaches for getting intra-dimensional patterns are memory-like and run outside of the database. On the other hand, when we deal with data warehouse the size of tables is extremely huge for memory copy. In addition, using a pure SQL-like approach is quite inefficient. Actually, those implementations rarely take advantages of database programming. Furthermore, RDBMS vendors offer a lot of features for taking control and management of the data. In section 2.2 we presented a pattern growth mining approach by means of database programming for finding all frequent itemsets. Thus, ***the main idea was to avoid one-at-a-time record retrieval from the database, saving both the copying and process context switching, expensive joins, and table reconstruction***. The empirical evaluation of our strategy shows that runs competitively when compared with most known itemset mining implementations based on SQL.

Classical association rules are by nature intra-transaction based, i.e., the associations occurs among the items within the same transaction. Further, those transactions are usually associated with a dimensional context which is typically ignored. This prevents that patterns, occurring along this dimension (i.e. associated context), are discovered. In section 2.3 we have introduced a strategy for the extraction and analysis of inter-transactional patterns. These patterns allow us to discover relationships that occur among transactions within a dimensional time frame. The method consists in the combination of association and sequence mining. First, association mining is applied to transform a dimensional transactional database into a sequence database. The latter is then mined in order to obtain frequent sequence patterns. Sequence patterns are augmented with additional distance (among events) information. ***We also have experienced that by using a measure like cvd allow us to reject patterns that presents significant distance variation, and thus, avoiding the representation of misleading patterns***.

Finally, inter-dimensional patterns fulfilling user's constraints (with relation to the measures of confidence, lift and others) are extracted.

Chapter 3

Aggregation-based Single-dimensional Mining

In this chapter we present several aggregation-based mining methods particularly devised for a real application scenario. We tackle the problem of mining superimposed fraud situations in telecommunication systems. Previously, in chapter 2 we have discussed count-based mining methods for getting intra and inter-dimensional patterns. Those count-based mining methods are quite constrained with respect to the aggregation function used, i.e., count ones. Thus, for handling more elaborated functions we need to design methods which can allow computation of complex aggregating functions such as, average and standard deviation. We propose three anomaly detection methods based on single-dimensional aggregations or just signatures. The first method relies on a signature deviation-based approach. The second one explores dynamic clustering analysis while the third method works towards community behavior. All methods are aggregation-based mining ones in sense that first proper aggregations are made over a particular dimension and then mining methods are applied to highlight interesting patterns. Experiments carried out with real data, voice call records from an entire week, corresponding to approximately 2.5 millions of CDRs and 700 thousand of signatures processed per day, allowed us to detect several interesting situations. Preliminary results and discussion with fraud analysts has already proved that those methods are a valuable tool to assist them in superimposed fraud detection. A preliminary version of this chapter were published in two papers presented in [Alves et al. 06] and [Ferreira et al. 06].

3.1 Single-dimensional Aggregations as Signatures

In superimposed fraud situations, the fraudsters make an illegitimate use of a legitimate account by different means. In this case, some abnormal usage is blurred into the characteristic usage of the account. This type of fraud is usually more difficult to detect and poses a bigger challenge to the telecommunications companies. Telecommunications companies use since the 90's decade several kinds of approaches based on statistical analysis and heuristics methods to assist them in the detection and categorization of fraud situations. Recently, they have been adopting the use and exploitation of data mining and knowledge discovery techniques for this task. In this chapter we focus on superimposed fraud detection in telecommunication systems. We first discuss two methods for discovering fraud situations through mining anomalous customers' behavior patterns. These methods are based on the concept of signature [Cortes & Pregibon 01], which has already been used successfully for anomalous detection in many areas like credit card usage [Kou et al. 04], network intrusion [Lunt 99] and in particular in telecommunications fraud [Cortes & Pregibon 01]. The main goal was established into two central statements:

- To detect deviating behaviors on useful time, giving better basis to analysts being more accurate in their decisions in the establishment of potential fraud situations.
- To elaborate aggregation-based mining methods which were capable of smoothing the course of dimensionality dilemma, as well as, revealing interesting anomalous patterns.

Our technique has as a core concept on the notion of signature. The curse of dimensionality is handled by signatures. Each signature profiles the user behavior from a single-dimensional point of view. Thus, we emphasize the work of Cortes and Pregibon [Cortes & Pregibon 01], since it was the main inspiration for the use of signatures. We have redefined the notion of signature. Thus, a signature of a user corresponds to a *vector of feature variables* whose values are determined during a certain period of time. A signature is a single-dimensional aggregation, in sense that all variables (or measures) are summarized according to a unique dimension. In this particular scenario of application the dimension to aggregate is the mobile phone number.

The variables can be *simple*, if they consist into a unique atomic value (ex: integer or real) or *complex*, if they consist in two co-dependent statistical values, typically the average and the standard deviation of a given feature. The choice of the type of the variables depends on several factors, like the complexity of the feature described or the data available to perform such calculation. A feature like the duration of the calls shows a significant variability which is much better expressed through an average(μ)/standard-deviation(σ) parameter. A feature like the number of international calls is typically much less frequent and thus an average value is sufficient to describe it. In table 11 we list the complete set of feature variables (fv) used in the context of this work. One can also see the signature model as a 1-D (one-dimesional) cube. Thus, the dimension to aggregate is the phone number and several measures are calculated according to table 11. This model is particularly different from the traditional data cubes where just few measures are associated and several dimensions are aggregated.

Description	**Type**
Duration of Calls	Complex
N. of Calls – Working Days	Complex
N. of Calls – Weekends and Holidays	Complex
N. of Calls – Working Time (8h-20h)	Complex
N. of Calls – Night Time (20h-8h)	Complex
N. of Calls to Diff. National Networks	Simple
N. of Calls as Caller (Origin)	Simple
N. of Calls as Called (Destination)	Simple
N. of International Calls	Simple
N. of Calls as Caller in Roaming	Simple
N. of Calls as Called in Roaming	Simple

Table 11. Description of the feature variables used in signatures and summaries.

The choice of the type of the variables depends on several factors, like the complexity of the feature described or the data available to perform such calculation. A feature like the duration of the calls shows a significant variability which is much better expressed through an average(μ)/standard-deviation(σ) parameter. A feature like the number of international calls is typically much less frequent and thus an average value is sufficient to describe it.

In Table 11 we list the complete set of feature variables (fv) used in the context of this work. A signature S is then obtained from a function φ for a given temporal window ω, where $S = \varphi(\omega)$. We consider a time unit, the amount of time in which the CDRs are accumulated and that in the end of this period are processed. A summary C, has the same information structure as a signature, but it is used to resume the user behavior in a smaller time period. Typically, a signature reflects the usage patterns for a period of a week, a month or even half year, whereas a summary reflects the periods of an hour, a half day or complete day. In this work, we considered the period of one day for a summary and a week for the signature.

3.2 Deviation-based

3.2.1 Evaluating Similarity of Simple Feature Variables

A simple feature is defined by a unique variable, which corresponds to the average value of the considered feature. For simple feature variable comparison we will make use of a ratio-scaled function. This type of function makes a positive measurement on a non-linear scale, which will be, in this case, the exponential scale. The used function is defined in the range [0, 1] and is defined according to the Equation 1:

$$d(S_x, S_y) = e^{-\{\frac{|S_x - S_y| \times B}{Amp}\}}$$

Equation 1. Similarity of simple feature variables.

In Equation 1, S_x and S_y are the two simple variables under comparison, B is a constant value, and Amp is the amplitude (difference between the maximum and minimum value) of the respective feature variable in all signatures space.

3.2.2 Evaluating Similarity of Complex Feature Variables

Complex feature variables are defined by two co-dependent variables. These variables correspond respectively to the average and the standard deviation of the considered feature. For two complex variables, $C_x = (M_x, \sigma_x,)$ and $C_y = (M_y, \sigma_y)$, the similarity function is defined in Equation 2, and is within the range [0, 1].

$$d(C_x, C_y) = d(S_x, S_y) \times \frac{|C_x \cap C_y|}{|C_x \cup C_y|}$$

Equation 2. Similarity of complex feature variables.

Equation 2 is the result of the combination of two formulas, the similarity function for simple variables (see Equation 1) and the ratio $\frac{|C_x \cap C_y|}{|C_x \cup C_y|}$. This ratio is also within the range [0, 1] and provides the overlap degree of the two complex feature variables by measuring the intersection of the intervals $[M_x-\sigma_x, M_x+\sigma_x]$ and $[M_y-\sigma_y, M_y+\sigma_y]$.

3.2.3 Evaluating the Distance among Signatures

Since the feature variables in the signature have different types, each variable has to be evaluated according to a distinct sub-function. Thus, the *dist* function is composed by the several sub-functions: *dist* = $\theta(f_1, f_2, \ldots, f_n)$. Consider as an example the simplification of a signature Sg = $\{(\mu_a,\sigma_a); \mu_b; \mu_c; (\mu_d,\sigma_d)\}$, where the first and the last feature variables are complex (calculated by Eq. 2) and the second and the third are simple (calculated by Eq. 1) variables. Let Sm = $\{(\mu'_a,\sigma'_a); \mu'_b; \mu'_c; (\mu'_d,\sigma'_d)\}$, be a summary. Since we are interested in considering deviation detection from a probabilistic point of view, i.e. the distance measure among two signatures Sg and Sm, would therefore correspond to the probability of Sm being different from Sg. The proposed distance function can be presented as:

$$D(Sg,Sm) = \sqrt{\alpha_1 \cdot f_1(Sg_1, Sm_1)^2 + \ldots + \alpha_n \cdot f_n(Sg_n, Sm_n)^2}$$

Equation 3. The distance among signatures.

Different distance functions can be provided, by the fraud analyst, by setting weighing factors α_i to different values. The use of different distance functions will allow detecting deviations in different scenarios. The overall distance function can be re-defined as in 2.

$$D(Sg,Sm) = Max\{dist_1(Sg,Sm), dist_2(Sg,Sm), \ldots, dist_m(Sg,Sm)\}$$

Equation 4. The maximum distance among signatures.

If according to the distance function, a threshold value ε defined by the analyst is exceeded, Dist(Sg, Sm) > ε, then an alarm should be raised to future examination of the respective user. Otherwise, the user is considered to be within its normal behavior.

3.2.4 Detecting Anomalies

The anomaly detection procedure based on signature's deviation consists in several steps. It starts by a *loading* step, which imports the information to a local database staging area. This information refers to the signature and summary information of each user. The signatures are imported only once, when the system is started. All the signatures of a user are kept through time. Such information will also be useful for posterior analysis. A signature may have two different status Active or Expired. For each user only one signature can have the Active state, and it is the most up to date one. The *processing* step corresponds to the algorithm described in [Ferreira et al. 06], and follows the previous equations for calculating the distance and similarities among signatures. According to equation (4), if an alarm is raised, the user is put on a *blacklist*. This is performed on the *triggering alarm* step, which is based on the calculation of the whole distance functions over the signatures. At the end, all the raised alarms have to pass through the analyst verification in order to determine if this alarm corresponds or not to a fraud situation. The evaluation of the alarms is supported by the interface of the system that employs features of dashboard systems, providing a complete set of valuable information.

3.2.5 Updating Procedure for Signatures

The updating process of the signatures follows the ideas presented in [Cortes & Pregibon 01]. The update of a signature S_t in the instant t+1, S_{t+1}, through a set of processed CDRs (summary) C, is given by the formula:

$$S_{t+1} = \beta.S_t + (1-\beta).C$$

Equation 5. Updating signatures.

The constant β indicates the weight of the new actions C in the values of the new signature. Depending on the size of the time window ω this constant can be adjusted [Cortes & Pregibon 01]. In contrast to the system in [Cortes & Pregibon 01], the value of signature is always updated. If the $Dist(S_t, C) \leq \varepsilon$ then the user is considered to have a normal behavior. If $Dist(S_t, C) > \varepsilon$ then an alarm is triggered, nevertheless the signature continues to be constantly updated. The reason for this is that the alarm still needs to pass through the analysis of the company fraud analyst. It could be the case in which the analyst considers it as a false alarm. The continuous update of that user signature avoids the loss of information that was gathered between the moment when the alarm was triggered and the moment the analyst gives his verdict.

3.3 Clustering-based

3.3.1 Dynamic Clustering of Signatures

The analysis of changes in the clusters topology over a period of time will provide valuable information for the better understanding of the usage patterns of the telecommunications services. ***In particular, the detection of abrupt changes in cluster membership may provide strong evidences of a fraud situation***. We propose the application of dynamic clustering analysis techniques over signature data. Our aim is that these changes will also provide evidences to fraud analysts for establishing potential fraud situations.

3.3.2 Similarity of Signatures

Signatures are composed of simple and complex variables. Traditional similarity measures, like for example Euclidean distance, Pearson correlation, Jaccard measure will not be applicable for signature comparison. Therefore, we need to devise a new similarity measure which will allow us to determine similarities among signatures. We define the similarity between two signatures as the combination of the variable similarity measures defined in section 3.1. For two signatures Sg and Sg', where Sg_i and Sg'_i are respectively the feature variable i of Sg and Sg', and for n possible variables, the similarity measure can be defined as in Equation 6.

$$D(Sg, Sg') = \sqrt{W_1 \cdot d_1(Sg_1, Sg'_1)^2 + ... + W_n \cdot d_n(Sg_n, Sg'_n)^2}$$

Equation 6. The signatures similarity.

D(Sg, Sg') $\in$ [0, 1] and W_i defines the weight of the feature and $\sum_{k=1}^{n} Wi = 1$.With this signature similarity measure, we can compare all signatures. This will provide a N x N matrix, that summarizes the similarities among the N signatures. The clustering solution can then be obtained by taking into account the previous calculated matrix as the input.

3.3.3 Clustering Migration Analysis

According to the moment of the week, different usage patterns can be found [Alves et al. 06][Ferreira et al. 06]. These usage profiles are provided by means of signature clustering analysis, according to the method describe previously in section 3.3. Therefore, for each day of the week a cluster topology is provided. This topology describes customers' usage patterns during that period.

Each cluster is described by the characteristics of its centroid. The centroid is defined as a signature. This allows making direct comparisons of the signatures and clusters centroid. The comparison is made according to the Equation 6. The signature assignment to the cluster is done by comparing each signature against each cluster centroid, and it is assigned to the cluster in which has the smallest distance.

3.3.4 Absolute and Relative Similarity

In order to make the comparison of signatures against cluster centroids, two types of similarity measures can be defined: absolute and relative similarity. Absolute similarity defines the similarity value between the signature and the centroid in a given time moment t. This value is calculated according to Equation 6. Relative similarity relates the absolute similarity between instant t and t+1, providing the percentage of the signature variation between two consecutive time instants. This value is obtained through the Equation 7:

$$\Delta = \{1 - \frac{D(S_i, SignCl[S_i]_{t+1}}{D(S_i, SignCl[S_i]_t}\} \times 100\%$$

Equation 7. the absolute and relative similarity of cluster signatures.

In Equation 7, S_i corresponds to a signature and $SignCl[S_i]$ to the cluster that S_i belongs in the moment t. Figure 9 shows a positive variation, where the signature S_i is closer to the centroid 0 in the instant t than in t+1.

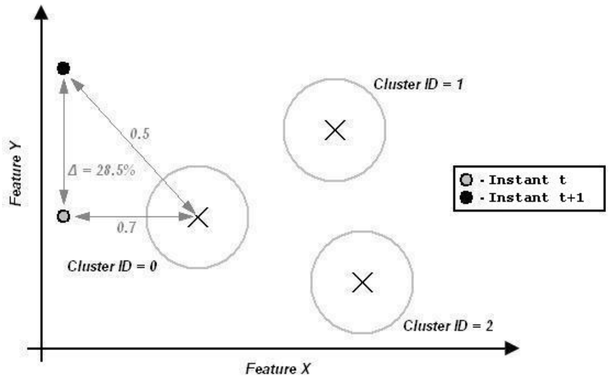


Figure 9. Positive variation of the relative similarity of the signature.

A negative value of the relative similarity in the instant t+1, indicates that the signature S_i is now close to the centroid of the cluster that it fits in the instant t. Nevertheless (see figure 10), we can detect to a cluster membership change, since now S_i is now closer to another cluster (cluster 1).

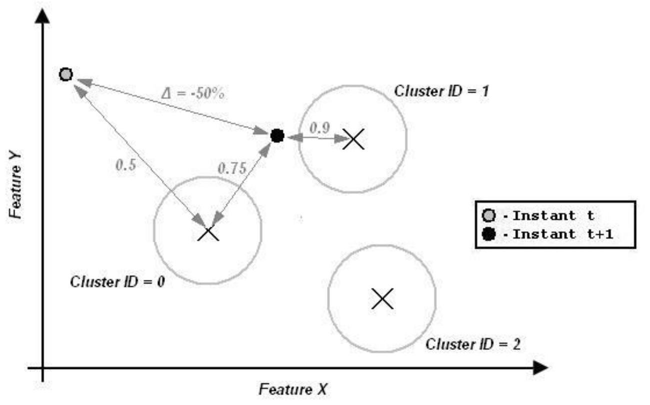


Figure 10. Negative variation of relative similarity of the signature and change cluster membership.

We define a *cluster membership change* as follows: a signature S changes its cluster membership to cluster C_j in the instant t+1, if it belongs to cluster C_i in the instant t, in the instant t+1 the distance $D(S,C_j)$ is minimal concerning all clusters and $D(S,C_j)_{t+1} < D(S,C_i)_t$. All the data relative to the cluster membership of the signatures are kept for posterior analysis. These data, which we call *Historical data*, will make possible to assess the evolution of the customer behavior through time. In order to offer the analyst a tool for a better examination of the changing behavior of the customers, during a defined interval, *analysis reports* can be generated [Alves et al. 06]. This tool will provide the identification of all the conditions used, as well as, the average and standard deviation of the signatures variations and the maximum, minimum and average values for all the signature feature variables. The deviating signatures detected are included into a *blacklist*, for further analysis.

```
dia2:   analysis1     -0.06889211    0.06934437
dia2    961994000     1      0      0.744   0.07    analysis1
dia2    961215540     7      0      0.658   -0.07   analysis1
...
```

Figure 11. Example of fraud situations in the blacklist.

Figure 11 shows an example of a fraud situation on a blacklist. The first line contains a header with the temporal reference, the analysis report description, and the limits of the range [μ-2σ, μ+2σ], which indicates that any variation outside this limit is considered an abnormal situation. For the next lines, it is listed the moment when the anomaly was detected, the signature identification (phone number), the cluster where the signature belongs, a flag indicating a cluster membership change (in the positive case), the absolute similarity of the signature and the cluster, the relative similarity (variation) and as the last column the description of the respective analysis report. More detailed information about the methods, as well as, scalability issues regarding its application can be obtained in [Ferreira et al. 06] [Alves et al. 06].

3.4 Evaluation Study: Deviation and Clustering-based Methods

In order to assess the quality of the first two methods in detecting anomalous behaviors, we have examined the data correspondent to a week of voice calls from a national mobile telecommunications network. By using this real dataset, we were able to evaluate not only the applicability of these methods but also the feasibility of using aggregation-based mining methods on a single level of summarization, and then delivering the detection of interesting fraud situations through adaptation of traditional mining methods over signatures.

The complete set of *Call Details Records* (CDR) corresponds to approximately 2.5 millions of records, and 700 thousand of signatures processed per day. Up to now, there isn't exists any accurate database with previous cases of fraud. Thus, the settings of our methods were guided by a small list of 12 customers (fraudsters in the referenced week), provided by the fraud analyst in order to detect other similar behaviors. In this first stage of detecting anomalous situations, we are interested on the effectiveness of our methods. Therefore, we worked on a subset of the previous data concerning to a sample distribution with approximately 5 thousands summaries per day and its respective signatures to the whole week. The detection process was carried out by applying the method described in section 3.2 and 3.3. Several thresholds (ε) were used and basically four main distance functions were designed combining different feature variables and weights. An illustration of the alarms generated by the deviation-based approach is given in Table 12. Pay attention to the most right (gray) column, further investigation on those alarms shown that some of them were real fraud situations.

ε/day	**0.8**	**1.0**	**1.2**	**1.6**	**2.0**
Tue	2141	649	139	50	25
Wed	3029	1145	251	103	56
Sat	1006	560	150	39	23

Table 12. Different thresholds (ε) and the alarms rose for three particular days of the week.

For getting more understanding under the circumstances in which those alarms where generated one must investigate the impact of each variable over the MAX distance function (see Equation 4). In figure 12 we exemplify such evaluation by allowing Top-K queries over the complete set of alarms. We also verified that the most 10 imperative anomalous situations were raging from 2.76 up to 3.33 concerning its distance function. The feature variable which has more impact over the distance calculation is the international call (originated ones). On the other hand, in Figure 13 we can see that workhours variable has great importance to the distance calculation over the whole period.

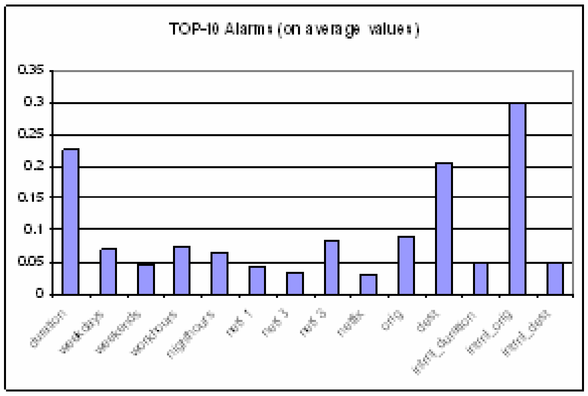


Figure 12. The impact of each feature variable over the top-10 higher alarms.

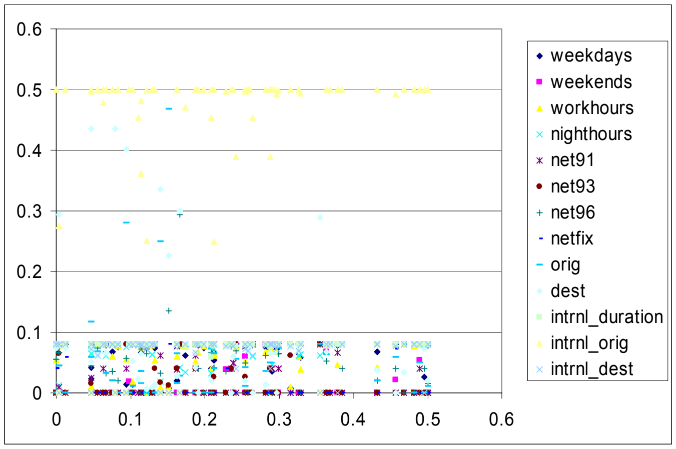


Figure 13. An overall picture of feature variable distribution over the max distance (ε ≥ 2).

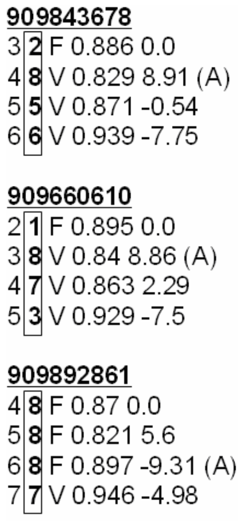
It is important to mention that both methods (i.e. deviantion- and clustering-based) provide just insights that could be recognized as anomalous situations. In fact, the characteristics of the data provided by the analysts don't allow us to apply any classification technique. Therefore, it is quite hard to evaluate, precisely, the rates for false positive and false negatives. Although, given the small list provided by analyst we can report a recall of 75%.

In order to complement the previous results we further make use of a dynamic clustering approach to detect suspect changes on cluster membership over the whole week. The identification of those changes will trigger alarms for future inspection. After several executions, the qualities of the clusters were maximized with 8 clusters. The distribution of the alarms raised by this method can be figured out in Table 13.

Cluster	**Tue**	**Wed**	**Sat**
1	3	9	1
2	9	7	123
3	3	12	71
4	5	17	16
5	23	21	22
6	20	31	40
7	8	11	26
8	52	72	0

Table 13. Alarms raised per cluster for three particular days of the week.

The bottom (gray) line in Table 13 shows the cluster with the highest number of calls. Figure 14 shows an example of changing on cluster membership, which represents a real fraud situation identified by this method. The first and second customers pass from a cluster (1 and 2) with a lower average of number of calls to the cluster with the highest number of calls (8), in days 4 and 3 respective. The third customer in this example, although always in the same cluster, has registered a significant variation between days between days 5 and 6.



```
909843678
3 2 F 0.886 0.0
4 8 V 0.829 8.91 (A)
5 5 V 0.871 -0.54
6 6 V 0.939 -7.75

909660610
2 1 F 0.895 0.0
3 8 V 0.84 8.86 (A)
4 7 V 0.863 2.29
5 3 V 0.929 -7.5

909892861
4 8 F 0.87 0.0
5 8 F 0.821 5.6
6 8 F 0.897 -9.31 (A)
7 7 V 0.946 -4.98
```

Figure 14. Example of anomaly situations regarded to the increase in the number of calls.

By using dynamic clustering we can now report a recall of 91%. As one can see this method is a little bit susceptible for detecting anomalous situations than the previous one. This is explained by the relative similarity measure (see Equation 7) which provides a fine tuning of the clustering migration method by exploring signatures relative variation over the time (whole week). Finally, the overlap rate of both methods corresponds to approximately 62% for the whole sample used, and 66% for the blacklist provided by the fraud analyst. Meanwhile, the remaining cases, other anomalous situations with the same behavior of the previous cases detected, are under inspection by the company analysts. Thus, the next efforts will be heading to the development of a database of fraud cases, as well as, an induction rule engine to help analyst on the evaluation of the alarms.

Concerning the scalability issues preliminary results showed to us that the most costly step is the calculation of the summaries and signatures. It requires several aggregations functions over CDRs records with the purpose of grouping information by each customer. At this time, this is done by several SQL scripts over a Microsoft SQL Server 2005. By the time that this information is available we can make use of each method discussed in this work without pre-defined order to detect anomalies. When dealing with such huge data we have realized that working with chunks of information (summaries and signatures) plus clustered indexes structures, it improves the processing time without losing quality of the results by at least one order of magnitude. On the other hand adds a new trouble, in sense that, when sliding the window from ω to ω+1 requires rebuilding of the all respective indexes.

Finally, in case of using dynamic clustering we have divide the original chunk of data D, into a set of partitions D'i, mutually exclusive, in order to make the processing of each partition feasible. After all partitions have been processed, the last step is to merge all the clustering information resulted from each chunk processed. The parameters that described the cluster topology obtained for each block are gathered in a unique set of D'f. These parameters are considered the data objects for further processing of the final K clusters obtained.

Previously we have presented two methods for detecting potential telecom fraud situations. Both methods rely on the concept of signatures to summarize the customer behavior through a certain period of time. In the first approach, the user signature is used as a comparison basis. A possible differentiation between the actual behavior (summary) of the user and its signature may reveal an abnormal situation. The second approach uses dynamic clustering analysis in order to evaluate changes on cluster membership over the time. The clear basis of these detection-based methods is that they complement each on reporting anomalous situations. For instance, in our evaluation study we got an overlapping of 66% fraud situations which was raised by the proposed methods. ***The experimental evaluation performed with data from a week of voice calls, and respective comparison, with a list of previously detected fraud cases, allowed us to conclude about the high rate of true positives (91%) detected by the proposed methods***. Additionally, they discovered other fraud situations which were not reported previously by the analysts. Preliminary discussion with fraud analysts gave us feedback about the promising capabilities of the proposed methodologies. These strategies clearly point out the importance and applicability of aggregation-based mining methods on a real application scenario.

3.5 Graph-based

As a general data structure, graphs have become increasingly important in modeling sophisticated structures and their interactions, with broad applications including chemical informatics, bioinformatics, web analysis and telecommunications. In this section we explore the application of graph mining for identifying anomalous situations on telecommunications call detail records (CDRs). This application is interesting, both because of its size and its rate of changes.

The main motivation is that those changes may reveal interesting situations of fraud, particularly in this work, superimposed ones. By using graphs, we can represent the dynamic interactions in mobile telecom networks with the whole set of call records, without losing information, and also providing a rich structure for revealing abnormalities. We used annotated digraphs to represent the communication networks. The dynamics of changes are modeled into the graphs by weighting factors using damping ideas. This representation highlights the fact that more recent CDR contributes more heavily to the graph at time t. Thus, new information is aggregated in accordance with this representation and evaluated through pre-defined recursive functions.

We also provide the exploration of several statistics over the graphs. Those statistics not only are important for querying graphs, but also play a key role to define anomaly functions for searching suspect situations (suspect graphs). We have designed functions to reveal suspect objects (subgraphs/vertices/edges). By manipulating those functions the fraud analyst can set up combined thresholds to points out suspect objects. Furthermore, those suspect graphs, can be matched against other graphs (which are known cases of fraud) in order to find out similarities with respect to its common networks. We called this operation as graph overlapping. Experimental evaluation carried out with real data, from a local mobile telecom company, showed us the feasibility and scalability of the proposed methods.

3.5.1 Motivation

As a general data structure, graphs have become increasingly important in modeling sophisticated structures and their interactions, with broad applications including chemical informatics, bioinformatics, web analysis and telecommunications. In this section we explorer the application of graph mining for highlighting interesting and anomalous situations on a mobile telecommunications scenario. Such anomalous situations may be indicative of fraud cases, and it may also isolate interesting network (possibly fraudulent) communities.

We propose a simple but effective approach for mining suspect (or deviating) call graphs. It is not our aim to compare the effectiveness of several graph mining methods for getting graph patterns. The call graphs discussed in this section are a representation of the social network behavior of several customers from a local mobile company[1]. Recently, we have also conducted other studies on telecom fraud detection. These works were concerned in understanding customer behavior using signatures [Ferreira et al. 06] and dynamic clustering [Alves et al. 06].

In section 3.4, telecommunications customers were analyzed on an individual basis. In this new study we extend previous works, combining individual customer call usage information with its respective call network behavior.

The main contributions proposed by this approach are:

1. A ***dynamic model*** for mining evolving call graph networks. The model is constantly updated when new information is available;
2. ***Anomaly functions***, so the graph can be inspected by employing customized statistical functions to detect significant deviations on graph components (subgraphs/vertices/edges);
3. ***Graph similarity evaluation,*** after identifying suspect graphs, fraud analysts should be able to evaluate those graphs against other ones representing previously detected fraudulent behavior.

3.5.2 Problem Definition

The problem tackled in this section can be stated as follows:

Based on information provided by the customer call detail records (CDRs), build an evolving model which helps fraud analysts to understand customer social network behavior – specially those ones concerning to potential fraud situations.

Several studies show that telecommunication companies lose large amounts of money every year due to a large diversity of fraudulent cases [Nanavati et al. 06] [Cortes et al. 03]. Besides, fraud is continuously evolving and telecommunications networks generate huge amounts of data

[1] This work was developed under the SFonTEL project. Its main goal is to employ mining methods to understand social network behavior from CDRs. It is a follow-up of FRATELO project, where several methods for fraud detection in mobile telecommunications systems were studied.

(sometimes of the order of several gigabytes per day) the detection and identification of fraud cases on useful time is extremely hard and costly. Among the many kinds of fraud in Telecommunications, we are particularly interested in the superimposed ones.

As described in previous sections, in superimposed fraud situations, the fraudsters make an illegitimate use of a legitimate account by different means. In this case, some abnormal usage is blurred into the characteristic usage of the account. This type of fraud is usually more difficult to detect and poses a bigger challenge to the telecommunications companies.

To build such fraud detection model, we generalize the method of Cortes et al [Cortes et al. 01] [Cortes et al. 03] using an annotated graph-based mining approach to highlight potential situations of fraud.

Definition 1 **(Annotated Graph)** An annotated graph is a digraph $G':=(V,A)$ with:

1. *V*, is a set of vertices or nodes,
2. *A*, is a set of ordered pairs of vertices, called directed edges,
3. *info*, is a set of information that describes the edge *A*. Each edge *a(x,y,info)* from *A* denotes a direct connection from x to y and contains a set of attribute values (*info*).

Definition 2 **(Weighted Information)** The proposed model is constantly updated with new information and a weighting factor θ (ranging from 0 to 1). For a time instant *t*, the new information g'_t is combined with the information contained in the graph G'_{t-1}, of the instant *t-1* (see Figure 15):

$$G'_t = \theta \cdot G'_{t-1} \oplus (1-\theta) \cdot g'_t$$

Equation 8. Updating factor for evolving graphs.

By employing definitions 1 and 2, the proposed model is able to support two important requirements in this kind of application: 1) ***dynamic, reflecting the temporal evolution of existing behaviors***; and 2) ***adaptative, incorporating new behaviors***.

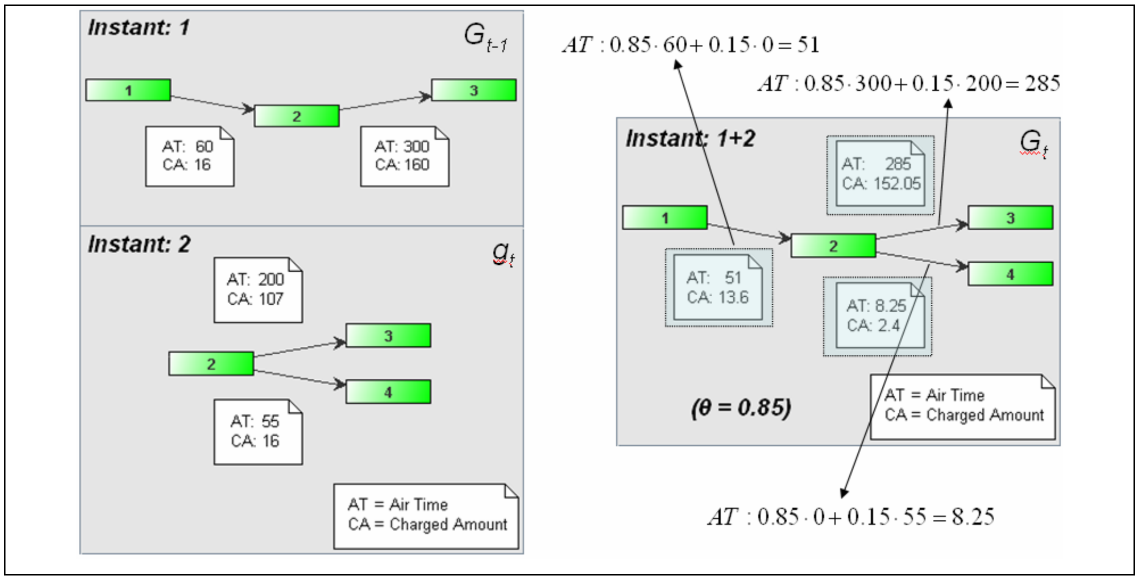


Figure 15. Illustration of the model evolution. Considering info attributes as air_time and charged_amount for each call and customer 1, 2, 3 and 4. The weighting factor was set to 0.85.

Due to operational limitations the updating process is made on daily basis. The setting values pointed by Cortes et al. in [Cortes et al. 01] have been used. Namely, *θ=0.85*; this corresponds to a situation where the graph reflects the call activity of the last month.

3.5.3 Graph Mining

With respect to our particular problem, we want to find graph patterns which point out anomalous situations. These anomalous situations should be indicative of potential fraud cases. In this sense, we are interested in finding abnormal behavior concerning subgraphs, nodes or edges. Different from [8] where deviations are evaluated from a pure power law distribution, we propose a more generalized approach finding significant deviations by exploring statistics over graph components (or objects). Therefore, our graph-mining approach attempts to detect suspect graphs, i.e., graphs which present significant changes on its vertex or edges. Furthermore, those suspect graphs, should be matched against other graphs (which are known cases of fraud) in order to find out similarities with respect to its common networks.

3.5.4 Finding Deviating Graphs

Suspect graphs are graphs in which its objects (or components) present significant deviations.

Definition 3 **(Suspect objects)** Are graph components in which present significant deviation when evaluated by the following ZScore function:

$$ZScore(Obj;i) = \frac{x_i - \bar{x}_i}{\sigma_i}$$

, where $i \in$ *info*, and Obj = {V, A}.

Equation 9. Score function for labeling deviating objects.

Since *info* should contains several attributes, deviations per vertex (customers) or edge (calls) are evaluated according to:

$$f(Obj;I) = \max(|ZScore(Obj;i)|)$$

, for all i $\in$ *I*, where *I=info* if Obj = A and *I={info}* if Obj=V. |*ZScore(Obj;i)*| is the modulus of the ZScore function.

Equation 10. Maximization of the score function for an object.

In this approach we restrict the evaluation of suspect graphs simply by scoring its vertices. In this sense, we rewrite equation 10 to:

$$f(Obj;I) = \max(|ZScore_{in}(V;i)|,|ZScore_{out}(V;i)|)$$

, where $ZScore_{in}$ and $ZScore_{out}$ denote the ZScore calculated for each vertex (customer), respectively for the received and originated calls.

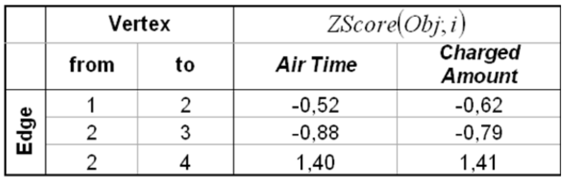
Equation 11. Maximization of the score funtion for Z_{in}-Z_{out} of an object.

Vertices are labeling according to the function f (see equation 11) as following:

1. Normal, if (f <= -1 and f > 0) or (f > 0 and f <= 1);
2. Suspect, if (f > -1 and f <= -2) or (f > 1 and f <= 2); and
3. Abnormal, if (f > -2 and f <= -3) or (f > 2 and f <= 3)

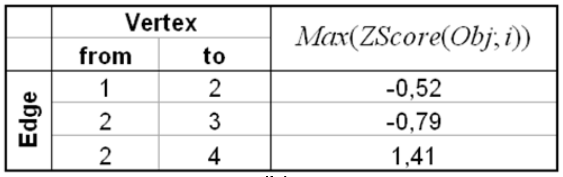
Vertex labeled as suspect or abnormal could be potential situations of either fraud or churning. Although in this work we are just interested on fraud situations. It is important to mention that the final decision, to evaluate whether a particular graph is a fraud case or not, should be made by the fraud analyst. This task of labeling suspect graphs only helps to filter out interesting cases for further analysis.

The next step should be to match this suspect graphs with other call graphs (previously identified as fraud situations), in order to evaluate similarities. Besides, fraud analysts could also make use of several statistics available to get deep analysis of all call graph network.



	Vertex		$ZScore(Obj;i)$	
	from	to	*Air Time*	*Charged Amount*
Edge	1	2	-0,52	-0,62
	2	3	-0,88	-0,79
	2	4	1,40	1,41

(a)



	Vertex		$Max(ZScore(Obj;i))$
	from	to	
Edge	1	2	-0,52
	2	3	-0,79
	2	4	1,41

(b)

Figure 16. It presents the results after applying equations 2(a) and 3(b) to detect suspect objects in graph G'_t of Figure 15.

3.5.5 Evaluating Deviating Graphs

We provide two alternatives for inspecting suspect graphs. The first one is to use graph statistics, so the fraud analyst can explore carefully all graph components (see Figure 16 and Figure 17). The second alternative is to evaluate the overlap degree between two graphs. The overlap degree between two graphs is given by a distance function, we called just graph overlapping.

Definition 4 **(Graph overlapping)** Given two vertices (A,B), their respective call networks can be compared by finding an overlapping degree between these two graphs:

$$overlap(A,B) = \frac{|\{Net(A)\} \cap \{Net(B)\}|}{Max(\{\deg(A)\} \cup \{\deg(B)\})}$$

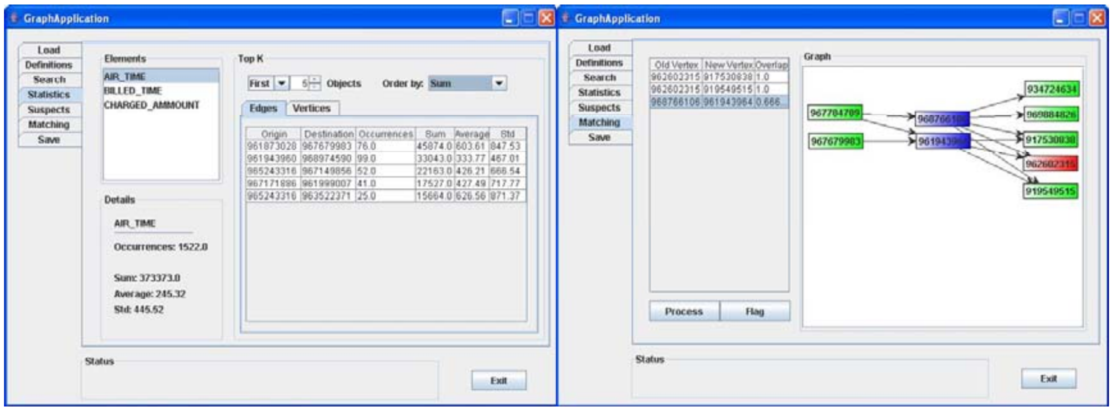
, where *Net(X)* corresponds to its adjacency list and *deg(X)* corresponds to the sum of the in-degree(X) + out-degree(X). |*Net(X)*| is the cardinality of a set *Net*(X).

Equation 12. The graph overlapping degree.

Since the overlap is between [0,1], the analyst can define a threshold (*γ*) to select the most similar networks. Consequently, if *overlap(A,B)* ≤ *γ* graphs are considered not similar, otherwise if overlap*(A,B)* ≥ *γ* they are similar. When *overlap(A,B)* = 1, we may say that both graphs are *isomorphic*. Thus, there exists a one-to-one correspondence between their node sets which preserves adjacency.

As a simple example, but not losing its generality, let's assume we want to evaluate the overlap degree between graphs G'_{t-1} and g'_t in Figure 15. We are interested to assess $overlap(G'_{t-1}(2), g'_t(2))$ for a threshold setting *γ*=75%. Thus, this ratio is given by evaluating $overlap(2,2) = \frac{|\{3\} \cap \{3,4\}|}{Max(\{2,2\})}$ which provides a final overlap of 50%. In summary, this evaluation should be carried out by pair comparison between each vertex (*suspect tag*) from G'_{t-1} against each vertex (*abnormal tag*) in g'_t.

In our application, normal vertices are green, suspect ones are yellow and abnormal ones are red. When vertices are selected for overlapping they appear on blue. Figure 17 shows a screen shot of the application.



Origin	Destination	Occurrences	Sum	Average	Std
961873028	967679983	76.0	45874.0	603.61	847.53
961943960	968974590	99.0	33043.0	333.77	467.01
965243316	967149856	52.0	22163.0	426.21	666.54
967171896	961999007	41.0	17527.0	427.49	717.77
965243316	963522371	25.0	15664.0	626.56	871.37

AIR_TIME
Occurrences: 1522.0
Sum: 373373.0
Average: 245.32
Std: 445.52

Old Vertex	New Vertex	Overlap
962602315	917530838	1.0
962602315	919549515	1.0
968766106	961943964	0.666

(a) (b)

Figure 17. Two screen shoots of the call graph application. Figure (a) presents the *top-5* objects according to a statistic value (*sum*) annotated into the graph. . Figure (b) highlights graph objects evaluated during an overlap operation.

3.5.6 Evaluation Study

The evaluation study was carried out with real dataset. This data corresponds to a week of voice calls from a Portuguese mobile telecommunications company. The complete CDRs have several variables, but specifically in this study the fraud analyst set up the following variables to build the call graphs: *a_number (in), b_number (out), air_time, billed_time* and *charged_ammount*. More discussion about variables related to this dataset could be obtained in previous works [Ferreira et al. 06] [Alves et al. 06]. An example of the degree distribution of the graph (~5 millions of vertices) for Thursday can be seen on Figure 18. The in-degree and out-degree distributions are closer to other distributions presented in [Chakrabarti et al. 06]. It also shows that most customers have similar calling behavior.

It is quite impractical to evaluate the proposed methods while looking at to the complete week dataset, given the huge size of these call graphs (see Figure 18). To make this experimental study feasible, the fraud analysts have selected a more reasonable week dataset for the running test. In this sense, the final dataset correspond to ~120K calls covering ~140k mobile phone numbers.

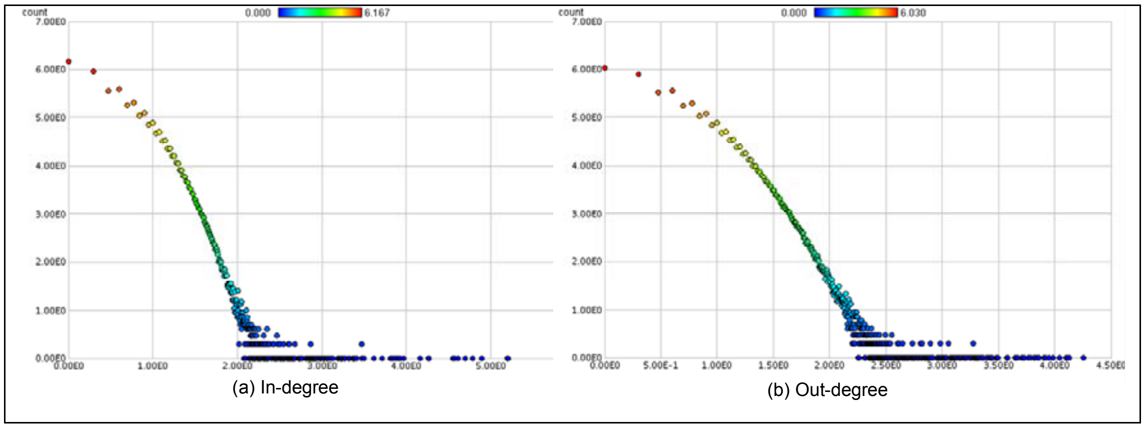
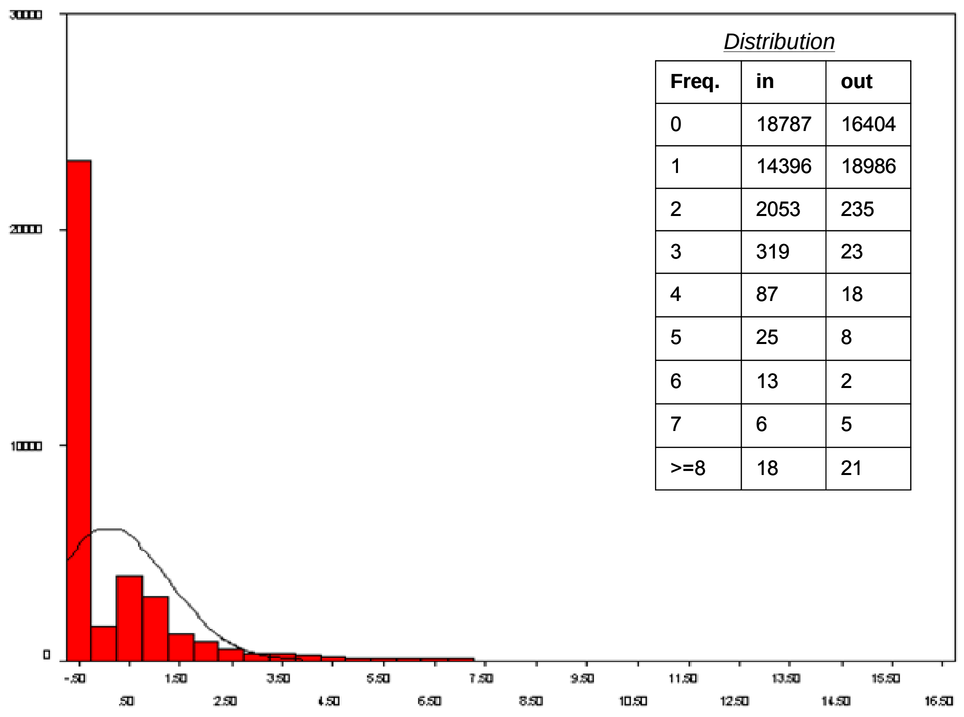


Figure 18. In and Out degree distributions for all calls made on Thursday.

The first aspect to be evaluated is the practical implications of the functions defined on section 3.5.4. Figure 19 shows the plot of the normal curve for equation 9. It corresponds to calls made on Thursday. Thursday was chosen since it is day of the week with larger activity. The ZScore values for this day are: σ=1.17, μ=0.11 and for a total of 35702 cases. It can be generally considered that customers have a behavior that is reflected through a normal distribution. A minority of the cases (7%) show a considerably deviating behavior.

The complete view of this labeling task for the entire week is illustrated on Figure 20 (b). As we would expect, the number of new vertices increase regularly when compared with the number of edges (see Figure 20 (a)). Although, a large number of potential fraud situations were triggered, preliminary feedback given by the fraud analysts has already proved that our methods are a valuable tool to assist them in fraud detection. To make this analysis more reliable we strongly advise using previous methods [Alves et al. 06] [Ferreira et. Al 06] to assess and to complement the quality of results achieved in this presented approach.



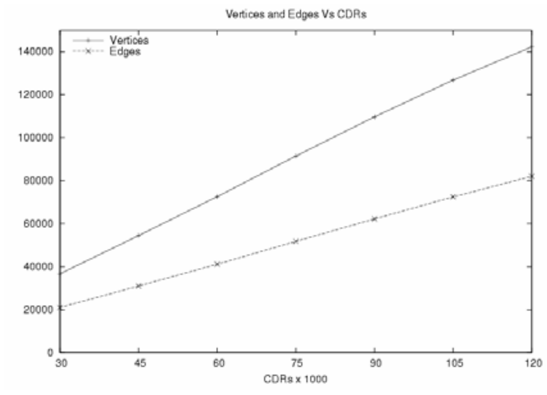
Distribution

Freq.	in	out
0	18787	16404
1	14396	18986
2	2053	235
3	319	23
4	87	18
5	25	8
6	13	2
7	6	5
>=8	18	21

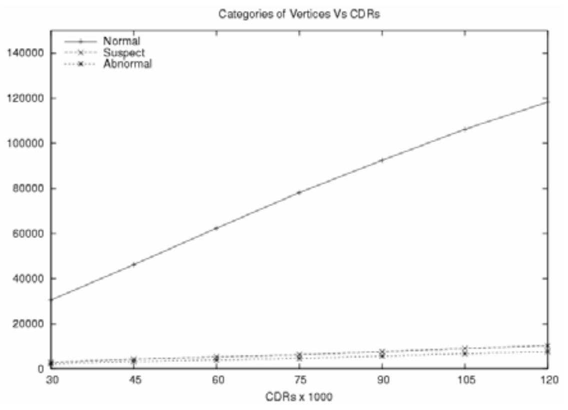
Figure 19. ZScore distribution for one week-day (thursday).

The second characteristic to be explored is the overlapping similarity between suspect graphs and abnormal graphs. Again, the main idea is highlight the most interesting cases for further analysis. Unfortunately, given the higher percentage of call graphs with low edge distribution the overlapping (suspect x abnormal) was not so effective on detecting similarities (usually just one edge associated). In addition, this pair-wise comparison has shown high computational costs. So, graph indexing and constraining should be taken place here to smooth this task. It also requires a well-made dataset to analyze not only the quantity but the quality of the overlapping task.

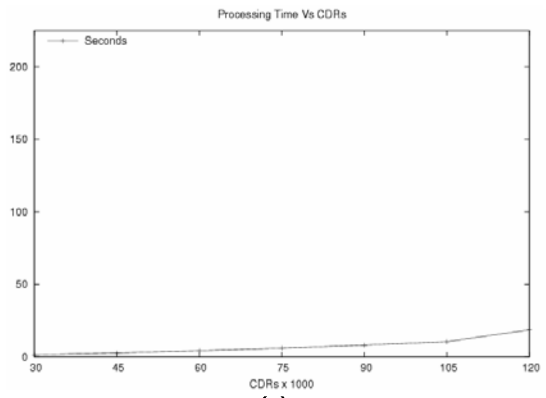
Scalability figures are presented in (Figure 20 (c) and Figure 20(d)). Those tests were performed on an Intel machine 3GHz with 3GB of main memory.



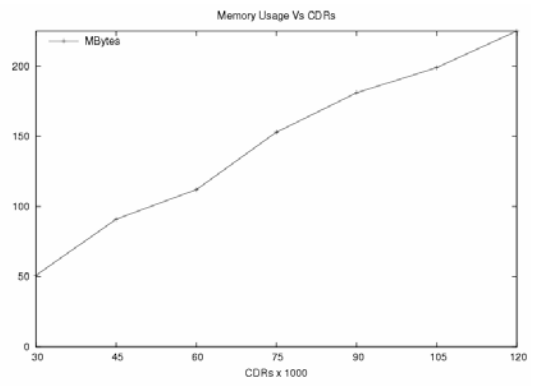
(a)



(b)



(c)



(d)

Figure 20. (a) Graph size (vertices and edges) w.r.t the number of CDRs; (b) Categories distribution (number of vertices) w.r.t number of CDRs; (c) Processing time for finding suspect graphs w.r.t the number of CDRs; (d) Memory usage w.r.t to number of CDRs.

3.5.7 Related Background

From the extensive literature in mining graph patterns, we emphasize two works that concern mining call graph patterns [Nanavati et al. 06] [Cortes et al. 03]. Recently, in [Chakrabarti et al. 06] was presented a complete survey about graph mining, covering algorithms, laws and generators. There also other kinds of graph patterns based on frequent patterns analysis [Han & Kamber 06]. With respect to graph mining on CDRs, Cortes et al [Cortes et al. 01] [Cortes et al.

03] propose a data structure based on the union of small subgraphs (*top-k edges*), called community of interests (COI) to handle large dynamics graphs. The previous methods do not explore deviations for detecting fraud situations. In [Pennock et al. 99] are presented an approach in which deviations are evaluated from a pure power law distribution. Our method is a generalization from the last mentioned works [Cortes et al. 03] [Pennock et al. 99]. Damping ideas for refreshing all call graphs, followed by exploration of deviating graph components have been used. Furthermore, we allow overlapping test for graph similarity.

3.5.8 Final Discussion

This work is part of the SFonTel Project, which relies on mining call graph patterns for better understanding of social mobile telecommunications networks. Particularly, we are interested on identifying deviating behaviors on graph components (subgraphs/vertices/edges). We have devised two methods for highlighting suspect graphs on a dynamic model. The first one is based on significant deviations of vertices and the later one related to graph overlapping. Before any calculation regarding the detection of anomalous graphs, aggregations are conducted over particular nodes, in this case, the phone number, and then deviating behaviors are identified. The main idea is quite related with the previously discussed methods, although the main difference is the way deviation is evaluated over those resulting aggregations.

Experimental evaluation carried out with real data from a local mobile company showed us the feasibility of the proposed methods. We verified that the majority of the call graphs present normal behavior and, as expected, just few customers show an abnormal behavior. Given that a higher percentage of call graphs has low edge distribution the overlapping (suspect x abnormal) was partially effective on detecting similarities. Furthermore, this pair-wise comparison has extremely costs. So, indexing and constraints should be considered in the future in order to smooth this task.

We have not explored graph patterns based on other metrics like hub and authorities, giant components, power laws and shortest-paths (small-word effect). New research will be heading to complement this study with these ideas in the near future.

3.6 Summary

In this chapter we have presented two methods for detecting telecom fraud situations based on the utilization of single-dimensional aggregations, i.e., signatures. Both methods rely on the concept of signature to summarize the customer behavior through a certain period of time. The clear basis of these detection-based methods is that they complement each other on reporting anomalous situations. For instance, in our evaluation study we got an overlapping of 66% fraud situations which was raised by the proposed methods. Additionally, they discovered other fraud situations which were not reported previously by the analysts. Preliminary discussion with fraud analysts gave us feedback about the promising capabilities of the first two proposed methods.

The third method discussed in this chapter relies on mining call graph patterns for better understanding of social network behavior. Particularly, we are interested on identifying deviating behaviors on graph components (subgraphs/vertices/edges). We have devised two methods for highlighting suspect graphs on a dynamic model. We experienced that the majority of the call graphs present normal behavior and, as expected, just few customers show an abnormal behavior. Given that a higher percentage of call graphs has low edge distribution the overlapping (suspect x abnormal) was partially effective on detecting similarities.

All methods presented previously encompass a set of single-dimensional mining methods which plainly confirm their usability, as well as, applicability on real data applications. The main advantage of such approach is to avoid high-dimensional aggregation by profiling with one dimension and then take the discovery of interesting patterns (anomalous situations) through adaptation of traditional mining methods over signatures. On the other hand, it should be also interesting to allow discovering from single to n-dimensional aggregation. The drawback of the single-dimensional approach is the extremely constraint of employing just one dimension to summarize multidimensional databases. Although, in the particular application scenario presented in this chapter was fairly useful.

Chapter 4

Aggregation-based N-dimensional Mining

Organizations have been used decisions support systems to help them to understand and to predict interesting business opportunities over their huge databases. Those huge databases are well known as *data marts* (DMs), organizing information and preserving its multidimensional and multilevel characteristics. A multidimensional data cube, also called cube, stores aggregation (or summarization) data for exploring interesting n-dimensional relations over DMs. OLAP tools have been used widely for retrieving information in a summarized way by employing customized cubing methods. The majority of these cubing methods suffer from being just data-driven oriented and not discovery-driven ones. In fact, real applications demand enhanced cubing methods which can employ mining functions into the discovery process. Other relevant aspect of DMs is the granularity of the information. By allowing concept hierarchy modeling, DM users can set the right level of information for mining interesting relations at any level of data abstraction. Finally, data marts grow quite fast, so an incremental OLAP mining process is a required and desirable solution for mining evolving multidimensional data.

In order to present a solution that covers all the previous mentioned issues, we propose a cube-based mining method which can compute an incremental cube, allowing concept hierarchy modeling, as well as, incremental mining of interesting relations by means of multidimensional and multilevel association rules. As far as we know, our method is the first one to support incremental OLAP mining over evolving DMs. The evaluation study using real and synthetic datasets demonstrates that our proposed method is a promising incremental cube-based mining method. A preliminary version of this study has been published in [Alves & Belo 07b].

4.1 Motivation

For a long time, organizations have been using decisions support systems to help them to understand and to predict interesting business opportunities over their huge databases. This interesting knowledge is gathered in such way that one can explore different what-if scenarios over the complete set of information available. Those huge databases are well known as data marts (DM), organizing the information and preserving its multidimensional and multilevel characteristics. OLAP tools have been used widely by DM users for retrieving summarized information, also called multidimensional data cube, through customized cubing algorithms. Since DMs are evolving databases, it is necessary to have the cube updated on a useful time. Usually, traditional cubing methods compute the cube structure from scratch every time new information is available. As far as we know, almost no one of them support an incremental procedure. Furthermore, those traditional cubing approaches suffer from being just data-driven oriented and not discovery-driven ones. In fact, real data application demands both strategies [Sarawagi et al. 98b].

Other relevant aspect of data cube exploration is the granularity of the information. When exploring the cube either is required detailed or general information. Concept hierarchy modeling comes as a required solution for setting the right level of information, providing an enhanced model for multidimensional and multilevel mining. On the other hand, handling a large number of dimensions and hierarchy's levels might be quite hard task for cubing methods. Therefore, bringing out some mining technique into the cubing process is an essential effort to reveal interesting relations on DMs [Kamber et al. 97] [Lu et al. 00] [Han et al. 99] [Alves & Belo 06].

In this chapter, we propose an incremental cube-based mining method for discovery-driven exploration of evolving cubes. We devised a new cubing strategy, capturing the goal of extracting interesting relations from data cubes on an incremental basis. Using this incremental cube, we develop a new cube-based mining method. The curse of dimensionality and granularity is handled by dimension selection during the mining process. More specifically, in this chapter, the main contributions can be summarized as follows:

- ***Incremental data cubing***. The cubing method proposed is inspired on a MOLAP approach [Zhao et al 97], and it also adopts a divide-and-conquer strategy. We've generalized bulk incremental updating from [Feng et al. 03]. Verification tasks through join-indexes are used every time a new cubing process is required. Thus reducing the search space and handling new information available.

- ***Multidimensional and multilevel mining***. Since the cube is processed from a DM. The implementation of hierarchies is supported by computing several cubes. The final cube is a collection of each processed cube. This requirement is essential to guide multilevel mining through dimension selection with the desirable granularity [Kamber et al. 00] [Han et al. 99]. Besides, it allows discovering interesting relations at any-level of abstraction from the cubes.
- ***Enhanced cube mining***. To discovery interesting relations on incremental basis, we support inter-dimensional [Lu et al. 00] and multilevel association rules [Kamber et al. 00]. We provide an apriori-based rule algorithm for rule discovering taking advantages of the cube structure, being incremental and tightly integrated into the cubing process. We also enhance our cube-based mining using other measure of interestingness [Alves & Belo 06].

4.2 Problem Formulation

Apart from the classical association rules algorithms, that usually take a flat database to extract interesting relations [Agrawal et al. 96], we are interested to explore multidimensional databases. In this sense, the data cube plays an interesting role for discovering multidimensional and multiple-level association rules [Kamber et al. 00].

A rule of the form X→Y, where body X and head Y consists of a set of conjunctive predicates, is a *inter-dimensional association rule* iff {X, Y} contains more than one distinct predicate, each of which occurs only once in the rule. Considering each OLAP dimension as a predicate, we can therefore mining rules, such as: Age(X, 30-35) and Occupation (X, Engineer) → Buys(X, laptop).

Many real data applications at mining associations require that mining be performed at multiple levels of abstraction [Han et. al 99]. For instance, besides finding in previous rule that 80 percent of people who age are between 30-35 and are Engineer may buy laptops, its is interesting to allow OLAP users to drill-down and show that 75 percent of customers buy macbook if 10 percent are Architect Engineer. The association relationship in the latter statement is expressed at lower level of abstraction but carries more specific and interesting relation than that in the former. Therefore, it is quite important to provide also the extraction of *multilevel association rules*.

Lets us now thinking in another real situation where the DM has been updated with new purchases or sales information. One may be interested to see if that latter patterns still holds. So, an *incremental procedure* is a fundamental issue on incremental OLAP mining of evolving cubes. We further present few definitions.

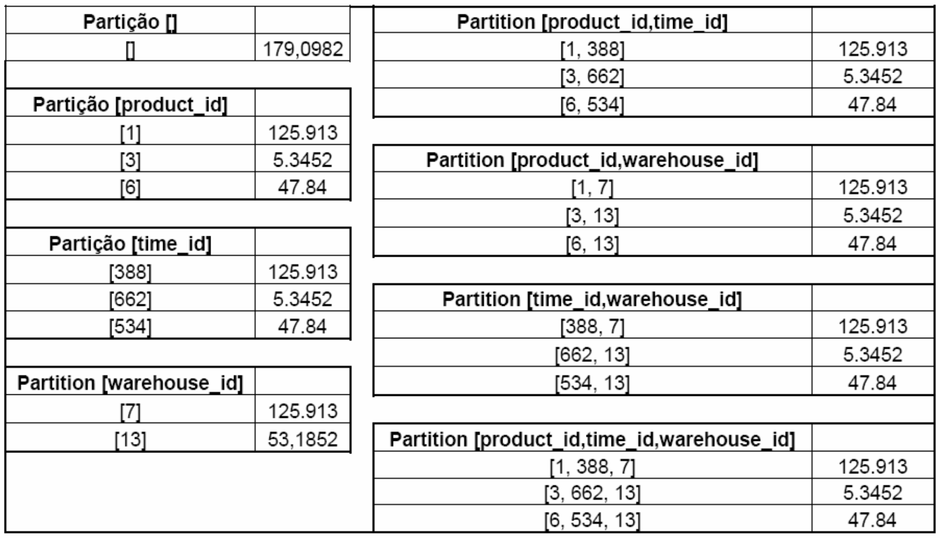
Definition 1 (**Base and Aggregate Cells**) A data cube is a lattice of cuboids. A cell in the base cuboid is a *base cell*. A cell from a non-base cuboid is an aggregate cell. An *aggregate cell* aggregates over one or more dimensions, where each aggregated dimension is indicated by a * in the cell notation. Suppose we have an n-dimensional data cube. Let i= (i_1, i_2, …, i_n, *measures*) be a cell from one of the cuboids making up the data cube. We say that i is an k-dimensional cell (that is, from an k-dimensional cuboid) if exactly k ($k \le n$) values among {i_1, i_2, …, i_n} are not *. If k = n, then i is a base cell; otherwise, it is an aggregate cell.

Definition 2 (**Inter-dimensional predicate**) Each dimension value (d_1,d_2,…,d_n) on a base or aggregate cell c is an inter-dimensional predicate λ in the form ($d_1 \in D1 \wedge \ldots \wedge d_n \in Dn$). The set {D1,…,Dn} corresponds to all dimensions used to build all k-dimensional cells. Furthermore, each dimension has a distinct predicate in the expression.

Definition 3 (**Multilevel predicate**) A Multilevel predicate is a *specialization* or *generalization* of an inter-dimensional predicate. Each predicate follows a containment rule such as $\lambda \in Di << Dj$, where << poses order dependency among levels (*i to j*) of abstractions for a dimension D.

Definition 4 (**Evolving cube**) A cube *C* is a set of k-dimensional cells generated from a base relation *R* on time instant *t*. For a time instant t_{n+1} all k-dimensional cells in *C* should be re-evaluated according to *R'* generating a evolving cube *C'*.

From the above definitions we can define our problem of mining evolving data cubes as: ***Given a base Relation R, which evolves through times instants t to t_{n+1}, extract interesting inter-dimensional and multilevel patterns DM, on incremental cubing basis.***



Partição []	
[]	179,0982

Partição [product_id]	
[1]	125.913
[3]	5.3452
[6]	47.84

Partição [time_id]	
[388]	125.913
[662]	5.3452
[534]	47.84

Partition [warehouse_id]	
[7]	125.913
[13]	53,1852

Partition [product_id,time_id]	
[1, 388]	125.913
[3, 662]	5.3452
[6, 534]	47.84

Partition [product_id,warehouse_id]	
[1, 7]	125.913
[3, 13]	5.3452
[6, 13]	47.84

Partition [time_id,warehouse_id]	
[388, 7]	125.913
[662, 13]	5.3452
[534, 13]	47.84

Partition [product_id,time_id,warehouse_id]	
[1, 388, 7]	125.913
[3, 662, 13]	5.3452
[6, 534, 13]	47.84

Figure 21. The complete set of partitions provided by the cube structure for Example 1

4.3 Cube-based Mining

4.3.1 The Cube Structure

The cube structure will be a set of arrays. Observing the cube as an object, it is simple to realize that the cube can be split into small cubes. These small cubes, usually defined as cuboids, are each one of the arrays (partitions) that together represent the final cube. The number of cuboids in the cube is defined by the number of elements of the power set of the dimensions to be processed, which is given by 2^n, where n is the number of dimensions. Inside the cuboids, each element is formed by a pair, where the left hand side is a set of dimensions and the right hand side is the measure (usually a numeric value) resulting from the aggregation of the data.

Example 1. Given the SQL-like notation for expressing a cube query is as follows:

```
Select  time_id, product_id, warehouse_id,
        sum(warehouse_sales) as sum
From inventory_fact_1997
group by product_id, time_id, warehouse_id
with cube
```

The final array is formed by a group of eight partitions. In Figure 21, we show just the first three tuples processed from the inventory fact table of the FoodMart Warehouse provided by Microsoft SQL Server.

4.3.2 Processing Cube

The cubing method consists of four steps. Basically, these steps can be divided as follows:

1. Read a line from the fact table
2. Combine the dimensions
3. Insert or (update) the data in the cube
4. Update the index

Step 1. The first step is reading a line from a fact table and identifying what are the dimension(s) and the measure(s).

Step 2. Next, the dimension(s) are combined using an algorithm based on a divide and conquer strategy, in order to get the dimensions power set, each of which is per-se a cuboid. The step 2 is further explained on next section. At this point, it is required to build the temporary data structure which is used additionally for inserting or updating the cube. It is important to mention that the cubing process supports the following distributive aggregate functions: sum, maximum, minimum, and count.

Step 3. Following step 2, the process checks whether the set of dimensions is already in the cuboid of the cube. In this case, the information is updated with the result given by the computation of the defined aggregating function. Otherwise, the data is inserted.

Step 4. Finally, the index is built so the access to the cube information can be done quickly.

4.3.3 Combining Cube's Dimensions

After reading a line from the fact table, it is necessary to combine the dimensions. In this step, the power set of the dimensions is generated. However, generating power sets usually requires a great amount of resources and time. Instead of using existing cubing strategies like top-down [Zhao et al. 97] or bottom-up [Beyer & Ramakrishnan 99], it was developed an algorithm based on a divide-

and-conquer (DAQ) strategy. These kinds of algorithms are recursive and proved to be extremely efficient among others algorithms, since it divides the initial problem in small ones until it reaches a trivial state which is straightforward to solve. Consequently, the algorithm combines the solutions until fulfils the goal. Besides, our proposed method supports an incremental procedure which is not allowed by the previous cubing approaches.

Before presenting the algorithm, first it is explained the two trivial cases.

1. Combine a singular set A of dimensions. In this case, since there is only one element in the set, then the result is the set itself. Example, X={A} gives [{A}].
2. Combine a set A with two dimensions. In this case, the result will is given by a set containing each element of A and the set A itself. Example, X={A,B} gives [{A},{B},{A,B}].

As it can be seen, the process has not taken into account the empty set, since it is not necessary at this time. Having presented the trivial cases, the remaining steps are discussed as follows. Suppose we have a set of dimensions D. The main goal is to combine the items of D among them, in order to get the power set of the items.

At the start, the set D is divided in two parts (D_1 and D_2). If the number of elements of D is odd, then D_2 will have one more elements than D_1. This step is executed again, recursively for each D_i until it reaches a trivial case. Whenever a set D_i is a trivial situation, then the combination is done according to the explanation given above.

The next step consists of combining the results of the trivial cases among them, until the final solution is achieved. After solving a trivial case, the result is a set as well. At this point, the algorithm takes two resulting sets that have the same ancestor and combine its results. To execute this, it is necessary to take into consideration the position of each element in the set. This is done as follows:

1. Put the same elements of both sets in a temporary set
2. Combine each element of each set, position to position
3. Add the result to the temporary set
4. Perform a left shift to the second set
5. Execute this step until the second set returns to its initial order

We demonstrate each step by a running example:

Given a set of dimensions to combine like X={A,B,C,D}. The temporary set Ts={{A},{B},{AB},{C},{D},{CD}} is built from the two resulting arrays A_1={{A,B},{A,B,AB}} and A_2={{C,D},{C,D,CD}}. After the combination step, and executing a left shift operation on A_2, the results are Ts={{A},{B},{AB},{C},{D},{CD},{AC},{BD},{ABCD}}, A_1={{A,B},{A,B,AB}} and A_2={{D},{CD},{C}}. Since the second set has not achieved its initial order, we repeat this combination-shift step. After more two rounds (*italic*, **bold**) we achieved the result set as Ts={{A},{B},{AB},{C},{D},{CD},{AC},{BD}, {ABCD},*{AD},{BCD},{ABC}*,**{ACD},{BC},{ABD}**}.

Applying the method (pseudo DAQ algorithm) explained above for the partition [product_id(0), time_id(1), warehouse_id(2)] = {6,534,13} from Example 1, we get a tree representation as follow:

```
{6,534,13}:[0,1,2]
|--{6}:[0]
    |--{6}:[0]
|--{534,13}:[1,2]
    |--{{534}:[1], {13}:[2], {534,13}:[1,2]}
|--{{6}:[0], {534}:[1], {13}:[2], {534,13}:[1,2], {6,534}:[0,1],…,{6,534,13}:[0,1,2]}
```

```
Pseudo DAQ Algorithm
(input: arraysOfDim[L[],I[]], output: powerset)
Combine array (L[], I[])
If sizeOf[L] = 1 or sizeOf[L]=2 //trivial case
    add pair (L,I) to powerset
If sizeOf[L]>2
    split L[] on left[] and right[]
    add each element of left[] and right[] to powerset
For each element in right[]
    For each element in left[]
         Let L[], I[]
         L[] <- getFirst(left[]), L[] <- getFirst(right[])
         I[] <- getSnd(left[]),   L[] <- getSnd(right[])
         add pair (L,I) to powerset
         right • leftShift(right)
Return powerset
```

Algorithm 3. Pseudo-code of DAQ algorithm for Cubing.

4.4 Enhancing Cubing

4.4.1 Incremental

While developing a process which was able to build a cube, the cube built could also be able to be incremented by an equivalent process or by the same process itself. Therefore, the process described in Section 4.3.2 is itself an incremental one. The proposed method runs line by line of the fact table. Therefore, it is only required to identify the new data in the fact table, and then start the process described in Section 4.3.3 followed with a checking task. This checking step is achieved by using a join-index guided by bulk incremental updating [Feng et al. 03].

Whenever data is inserted in that fact table, it is appended at the bottom (for instance on Oracle Databases, we can look for higher UUIDs), thus it becomes simple task to pinpoint the new lines, by saving the number of the lines that had already been evaluated.

4.4.2 Hierarchies

The implementation of hierarchies in the cubing process is handled by constructing several cubes instead of only one. Therefore, the final cube will be a collection of each cube processed.
Let the hierarchies be defined as follows:

$H_1 = \{L_{1,0}, L_{1,1}, L_{1,2},...., L_{1,p}\}$

$H_2 = \{L_{2,0}, L_{2,1}, L_{2,2},...., L_{2,q}\}$

....

$H_n = \{L_{n,0}, L_{n,1}, L_{n,2},...., L_{n,r}\}$

Where H_i is an hierarchy and $L_{i,j}$ is the jth level of hierarchy i. Thus, the number of cubes to be built is given by Equation 1.
Number of Cubes = $(p+1) * (q+1) * \dots * (r+1)$

Equation 13. The total number of cubes given n-levels of hierarchies.

As explained in Section 4.3.2, the cube is processed from a fact table. Although the cubing process will be the same, the table used to read the data will be slightly different. Instead of having only the fact table's fields, it will have some extra fields and some others from the levels of the hierarchies, which will substitute the fields of the corresponding dimension later on. Therefore, it will be required to construct several tables, one corresponding to each cube. These tables will be given by the combination of each and every level of the hierarchies. Once again, the algorithm described in Section 4.3.3 is called to combine dimensions in all hierarchies' level.

We can summarize the proposed method with hierarchies as follows:

1. First it is required the number of cubes to be processed
2. Then, the tables are generated by combining the levels of the hierarchies, using the algorithm described in Section 4.3.3. At this point, for each built table, apply the process describe in Section 4.3.2.
3. Finally, it is obtained a collection of cubes that all together form the final cube.

We provide a running example to show the total number of cubes to be calculated. Let's use Table 14 providing the dimensions and levels to compute.

	Dimensions		
Hierarchy level	*Product*	*TimeByDay*	*Warehouse*
0	ProdId	timeId	WarId
1		DayOMonth	WarName
2		Month	WarCountry
3		Year	

Table 14. Example of hierarchy levels for three dimensions.

The final solution is obtained by multiplying all dimensions' levels (Product * TimeByDay * Warehouse = 1 * 4 * 3 = 12 cubes). Finally, the number of tables is provided as follows.

Table 0 = {product_id, time_id, warehouse_id}

Table 1 = {product_id, day_of_month, warehouse_id}

Table 2 = {product_id, the_month, warehouse_id}

Table 3 = {product_id, the_year, warehouse_id}

Table 4 = {product_id, time_id, warehouse_name}

Table 5 = {product_id, time_id, warehouse_ctry}

Table 6 = {product_id, day_of_month, warehouse_name}

Table 7 = {product_id, day_of_month, warehouse_ctry}

Table 8 = {product_id, the_month, warehouse_name}

Table 9 = {product_id, the_month, warehouse_ctry}

Table 10 = {product_id, the_year, warehouse_name}

Table 11 = {product_id, the_year, warehouse_ctry}

4.5 Mining from Cubes

4.5.1 Extracting Inter-dimensional and Multilevel Patterns

The main goal of this process is the extraction of interesting relations from a cube. This extraction is guided by an Association Rule Mining approach. Association Rules (AR) can extract remarkable relations, as well as, patterns and tendencies [Han et al. 99]. However, it can also return a great amount of rules, which makes difficult either the analysis of the data or its comprehension. In addition, traditional frequent pattern mining algorithms are single-dimensional (intra-dimensional) in nature [Lu et al. 00], in sense of exploring just one dimension at a time. Besides, doesn't allow the extraction of inter-dimensional and multilevel association rules.

To enhance association rule mining from cubes, the discovery step is integrated with cubing. Thus, it allows getting inter-dimensional and multilevel association rules by using the intrinsic properties provided by the cube structure.

The rule extraction devised is based on Apriori-implementation [Agrawal et al. 96] and the most costly part, getting the frequent itemsets, is achieved by the proposed cubing algorithm. Each cuboid cell, along with its aggregating measure in the data cube, could be evaluated as a candidate itemset (CI). It is important to mention that by evaluating aggregating measures as CIs we are concerning to the population of facts rather than the population of units of measures of these facts. Thus, support and confidence of a particular rule is evaluated according to users' preference for a specific aggregating function (count, min, max, sum, etc…). ***It is clear that the cube structure provides a rich model for rule analysis rather than classical count-based analysis of ARs***.

Having found all interesting cuboids, the next step is generating rules. This implies firstly the calculation of the confidence of the rule for each non-singular itemset and check whether it satisfies the minimum pre-defined threshold or not. To generate a rule it is necessary to go through the following algorithm.

```
Pseudo Rule Algorithm
(input: cube, output: rules)
For each non-singular cuboid cell T
  Obtain the aggregate value of T (supOfT)
  Generate all the non-empty cuboid Si of T, using the algorithm
   described in Section 2.3
For each cuboid Si.
  Obtain the aggregate value of Si (supOfSi)
  Calculate the confidence (conf = supOfT/supOfSi)
If conf satisfies the minimum threshold ε
  Construct rule (Si → (T - Si))
If the rule is valid then store the rule
```

Algorithm 4. Pseudo-code of the rule algorithm for mining from cubes.

To verify the validity of a rule it is necessary to check if all the subsets (cuboids) of the rule are a valid rule [Agrawal et al. 96]. In other words, it is necessary to generate all the subsets Ri of Si and check whether Ri → (T - Si) already exists. If that is true then Si → (T - Si) is a valid rule. Otherwise, the rule must not be stored. One particular aspect that needs to be mentioned is the fact that the process must generate the rules starting with the cuboids that have fewer dimensions and finishing with the cuboids that have more dimensions (k-d cells).

Inter-dimensional association rules. Given the process explained previously, one just needs to select cube dimensions to generate valid multidimensional rules. For instance, Age(X, 30-35) and Occupation (X, Engineer) → Buys(X, laptop), it is a inter-dimensional rule. Furthermore, it allows slicing the cube to get valid rules for different cube's scenarios (sizing the maximum number of predicates to evaluate).

Multilevel association rules. One can also use the same process for getting multilevel association rules. Although, the granularity is controlled according to the set of pre-defined hierarchies (selecting dimensions) explained in 5.4.2. In this sense, one can explore several levels of interests in the cube to obtain interesting relations. Thus, the process is guided by roll-ups and drill-downs operation over the cube. For instance, Age(X, 30-35) and Occupation (X, Computer Engineer) → Buys(X, macbook), it is a multilevel rule (specialization of the multidimensional rule example above).

Even by using a cube-based mining (relying on support/confidence) approach the number of valid rules is still large to analyze. Furthermore, *what are the right cube abstractions to evaluate?* ***It might not be the case of materializing all cells to see that just a few really matters***. Thus, we smooth this problem by generating interesting rules according to the *maximal-correlated cuboid* measure of a cuboid cell [Alves & Belo 06b]. We further explore this measure with more details on chapter 5.

4.5.2 Enhancing Patterns Discovery

To avoid the generation of non-interesting relations we augment cuboid cells with one more measure called the maximal-correlated value of a cuboid cell. This measure is inspired on *all_confidence* measure, which has been successfully adopted for judging interesting patterns in association rule mining, and further exploited in [Alves & Belo 06b] for correlation and compression of cuboid cells. This measure discloses true correlation (also dependence) relationship among cuboids cells and holds the null-invariance property. Furthermore, real world databases tend to be correlated, i.e., dimensions values are usually dependent on each other. This measure is evaluated according to the following definition.

Definition 5 (**Correlated Value of a Cuboid Cell**) Given a cell c, the correlated value 3CV of a cuboid V(c) is defined as,

$$maxM(c) = \max \{M(c_i) | \text{for each } c_i \in V(c)\}$$

Equation 14. The correlated value of a cuboid cell, (Part I).

$$3CV(c) = M(c) / maxM(c) \quad \text{Eq.(3)},$$

where $M(c_i)$ corresponds to the aggregating value of this cell c.

Equation 15. The correlated value of a cuboid cell, (Part II).

By using the above definition we are able to find infrequent cuboid cells that may be interesting to the user but should not be obtained when using other measures of interests such as support, confidence lift, among others. Furthermore, given a set of abstraction levels to mine as interesting predicates, this measure will highlight most interesting multilevel patterns among those available predicates. In our evaluation study, we experienced the effects of correlated cuboid cells when using those measures of interest.

4.5.3 Incremental Patterns Discovery

The incremental rule discovery is achieved by combining NFUP approach [Chang et al 05] with the incremental cube maintenance (see next section). The NFUP algorithm relies on Apriori and considers only these newly added transactions. Since the cubing method is devised on incremental basis, it is simple task to identify cuboid cells that are still important to the rule discovery process.

4.6 Evaluation Study

The complete set of tests was elaborated in a PC Pentium 4 3.0 GHz, with 1 Gb of RAM, and Windows XP. The main code was developed with Java 1.5. We have elaborated two studies with the proposed method. In the first one, we evaluate the incremental feature in presence of different cubing strategies versus re-computation. The cubing strategy DAQ was compared with one bottom-up (BUC) [Beyer and Ramakrishnan 99] and other top-down (MOLAP) [Zhao et. al 97] approaches. Those strategies were chosen in sense that they present the most known cubing implementations. Both algorithms were implemented to the best of our knowledge based on the published papers. The second study evaluates the cube-base mining capabilities by incrementally maintenance of multidimensional and multilevel rules.

4.6.1 Incremental Data Cubing

In order to execute the tests for this study, it was developed a dataset generator to provide DMs in a star scheme model. Therefore, it is required to give as parameters the number of dimensions, the number of lines of the fact table, the number of lines of each dimension table, the number of levels of each hierarchy and the number of distinct elements in each column of the dimension tables. We have used three synthetics(S) DM, plus two real datasets(R). The real ones were the (F)oodMart DB provided by Microsoft SQL Server and the (W)eather Dataset (http://cdiac.esd.ornl.gov/ndps/ndp026b.html) for September 1985. The main characteristics of those datasets are presented in Table 15.

DS	S/R	FactSZ	NDims	HLevels
1	S-1	100,00	4	3L
2	S-2	250,00	5	4L, 3L
3	S-3	500,00	6	3L, 2L
4	R-F	164,558	5	3L, 2L
5	R-W	1,015,367	9	0L

Table 15. Characteristics of five data marts used in the evaluation study.

Having generated those datasets (S-1 to S-3), we also set a degree of updating (dg%), meaning insertions on fact tables. Thus, it was possible to measure the effects of each cubing strategy either by computing every fact table line-by-line from scratch or by computing it incrementally.

Figures 22(a), 22(b) and 22(c) illustrate the effects of DM updating x (re)cubing over datasets S-3, S-1 and S-2, respectively. We measure the speedup (processing time) ratio between (rc=re-compute cube) and (ic=incremental cube). This ratio (ic/rc), allows measuring the degree of improvement that ic achieves over rc. Although, we are working on extending the method to work with complex aggregating functions, in those experiments, just distributive ones were taken into account.

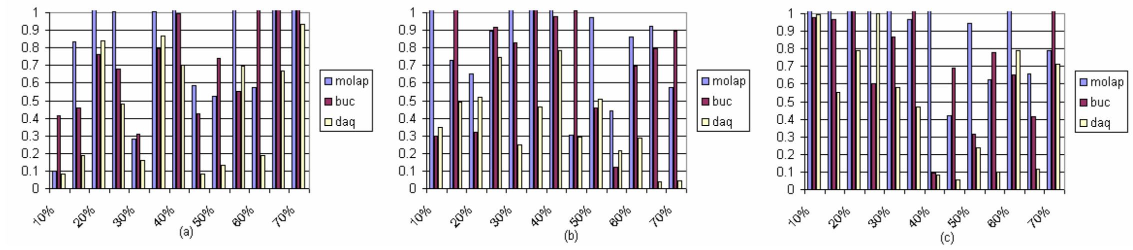


Figure 22. Shows the updating effects (dg%, x-axis) against cubing speedup(ic/rc, y-axis).

The Figure 22(a), 22(b) and 22(c) allows observe that in general ic performs much better than rc, specially when the degree of updating is low. We also notice that the speedup increases with respect to the size of the DM. Furthermore, one can see that DAQ generally shows better computational costs rather than other strategies.

4.6.2 Cube-based Mining

We have elaborated several experiments (observe the running time on ms) based on the value of the minimum support (%) with respect to different settings of predicates (see Figure 24). We also have constrained the maximum number of levels to explore as 3, and the maximum number of dimensions to 4. Figure 24(a) and 24(c) show the performance figures of mining interesting relations through different thresholds using R-F dataset. We can see that increasing the number of hierarchies (levels) doesn’t imply high costs on our incremental OLAP Mining. The other way around, increasing the number of dimensions plays few overheads. Although, we can also save cube computation when exploring the downward property of 3CV measure (Figure 23).

Performance figures with R-W dataset are given in Figure 24(b) and Figure 23(b). Again, we can see the effects of mining correlated cuboids when evaluating interesting relations on each dataset. The tradeoff between cube-based mining using confidence (conf) and correlated-cuboids (3CV) are evaluated according to low and higher thresholds. Low 3CV values will provide interesting relations regardless the high overhead provided by the conf measure. We can evidence this tradeoff while OLAP Mining datasets R-F and R-W on Figure 23(a) and Figure 23(b), respectively. Finally, we are able to evaluate our method with respect to update effects on a fact table. *The update effects generated from S-1 dataset for a dg%=70% (Figure 9(b)) is quite exponential when OLAP Mining two or three predicates with conf as measure of interest, but is quite stable when using 3CV.*

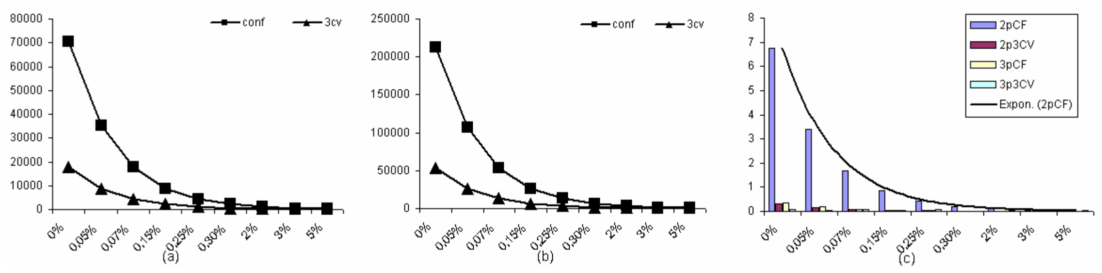


Figure 23. (a) and (b) present OLAP mining performance (runn.time(s), y-axis) for different measures of significance (%, x-axis). (c) presents the number of interesting cuboids (cb*k, y-axis) for different thresholds.

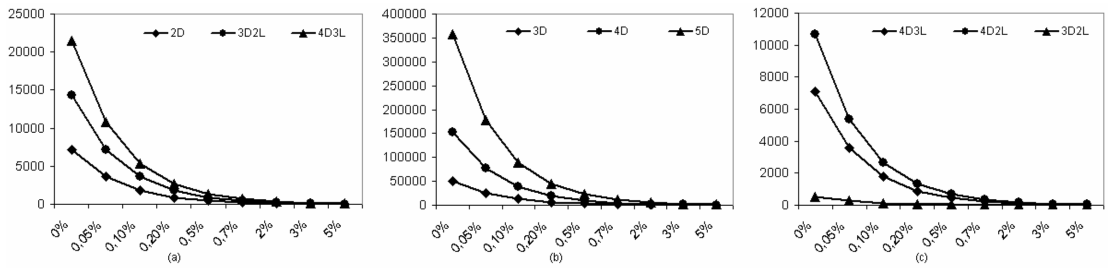


Figure 24. Illustrates OLAP mining performance (runn.time(s), y-axis) when getting different patterns through different support thresholds (%, x-axis).

4.7 Final Discussion

4.7.1 Related Work

OLAP has been taken attention of several researchers since the introduction of the cube operator [Gray et al. 96]. There are several works on data cube computation. The closer work with our method is the metarule-guided mining approach [Kamber et al. 97] [Han et al. 99], but incremental cubing issues were not mentioned in this study. To situate our work, among those cubing strategies, we first categorize them in three possible spots. The first one deals with data cube computation, and its many ways of cubing [Xin et al. 06] [Zhao et al. 97] [Beyer & Ramakrishnan 99]. The second spot is made of the several studies which try to integrate interesting exploitation mechanisms either by using statistical approaches [Sarawagi et al. 98b] or by integrating mining in the cube process [Alves & Belo 06]. The latter spot deals with OLAP over evolving DMs. Thus, those methods basically must deal with incremental issues either when cubing [Feng et al. 03] or when mining evolving databases [Chang et al. 05]. Given such simple categorization we can say that:

- w.r.t. ***first spot***. We compute the full cube, employing a DAQ strategy. One can also make use of equivalence class from [Xin et al. 06] [Alves & Belo 06] to explore cube semantics and compression. Our cubing method showed to be quite effective through our performance studied in presence of evolving data rather than other cubing strategies.
- w.r.t. ***second spot***. We use multidimensional association rules as our basis to explore interesting relations from DMs. Given that traditional association rules usually work on single flat table, our method uses cube structure to extract inter-dimensional and multilevel patterns on incremental basis. Furthermore, we have enhanced this mining task extending the proposed model for exploring correlated cuboid cells.
- w.r.t. ***third spot***. We provide a cubing method which is per-se incremental, and it can handle effectively the addition of new information on DM fact tables; it also provides concept hierarchy modeling, as well as, integrated and incremental multidimensional and multilevel mining.
- With the above discussion, our method is the first one to integrate all those important requirements in only one strategy for incremental OLAP Mining over evolving DMs. *An interesting topic for further research is to perform a detailed comparison with other*

cube-based mining methods, including updates caused by tuple deletion and in-place value modification.

4.7.2 Indexing

The index strategy adopted by our method is a join-index on the combined (foreign key) FK columns in the fact table. We provide an index for each partition (see Figure 21). In fact each cuboid has an index associated to. Therefore, if it is necessary to insert or update any information in a cuboid, first it is necessary to check if the information is already in the cube. Thus, firstly the index is analyzed to verify if there is any set of items that begins with the first item of the set of items of the information. If that is not true, then the information is not available in the cuboid and it can be inserted. Otherwise, the index will return a set of lines where the information is possibly stored. At this point, the process will go through the whole process discussed in section 4.5. If it is found, then the measure is updated based on the aggregation function used. Otherwise, the information is inserted. Figure 25, shows the effects of processing (DAQ) speed up while indexing datasets through different updating scenarios. The little overhead in Figure 25(b) is given by its multilevel properties (ranging from 3 to 4).

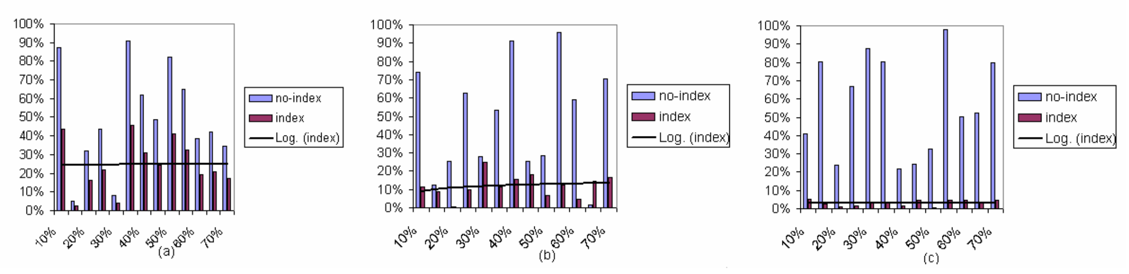


Figure 25. It shows the update effects (dg%, X-Axis) against index speedup of DAQ (Y-Axis) for datasets S3(a), S2(b) and S1(c).

4.8 Summary

We have presented an incremental OLAP mining method to explore evolving DM. The updating process on real DM applications implies that, the summarization made by cubing methods need to handle, with some incremental facility, the new information available. Furthermore, mining mechanisms should be integrated into the cubing process to improve the exploration of interesting relations on DMs either before of after OLAP. ***Our evaluation study demonstrates that our***

method is quite competitive and effective w.r.t. other known cubing strategies, also providing an integrated and incremental cube-based mining method for discovering knowledge over evolving DMs. We have also demonstrated that exploring correlated cuboids during the whole discovery process we are able to provide an essential tradeoff between processing time and accuracy patterns. Further, we evidence that mining multilevel patterns are less costly than inter-dimensional ones.

Chapter 5

Advanced Aggregation-based Mining

In this chapter we cover two aggregation-based mining methods devised for MDA in high multidimensional space. The first method effectively reduces the complete set of cuboids cells introducing a new notion called maximal-correlated cuboids cells which alleviates the curse-of-dimensionality problem. Through this cubing method one may find the maximal combinations among the grouping attributes and also keep its exceptional correlated cuboids, which can also indicates interesting changes on the cuboids during the cubing process. The second method studied issues and mechanisms on effective mining of Top-K multidimensional gradients. Gradient are interesting changes in a set of measures (aggregates) associated with the changes in the core characteristics of cube cells. We have also proposed a gradient-based cubing strategy (TopKgr-Cube) to evaluate interesting gradient regions in MDS. This strategy relies on cubing gradient regions which show high variance. Thus, the main challenge is to find maximum gradient regions (MGRs) that maximize the task of mining Top-K gradient cells through a set of Top-K probe cells. Preliminary versions of this chapter were published in two papers presented in [Alves & Belo 06b] and [Alves & Belo 07b] with respect to correlation and top-k gradient analysis respectively.

5.1 Aggregating Maximal-correlated Cuboids

The main idea of iceberg data cubing methods relies on optimization techniques for computing only the cuboids cells above certain minimum support threshold. Even using such approach the curse of dimensionality remains, given the large number of cuboids to compute, which produces, as we know, huge outputs. However, more recently, some efforts have been done on computing only *closed cuboids*. Nevertheless, for some of the dense databases, which are considered in this section, even the set of all closed cuboids will be too large. An alternative would be to compute only the *maximal cuboids*. However, a pure maximal approaching implies loosing some information, this is one can generate the complete set of cuboids cells from its maximal but

without their respective aggregation value. To play with some loss of information we need to add an interesting measure, that we call the ***correlated value of a cuboid cell***.
In this section, we propose a new notion for reducing cuboids aggregation by means of computing only the *maximal-correlated cuboids cells*, and present the M3C-Cubing algorithm that brings out those cuboids. Our evaluation study shows that the method followed is a promising candidate for scalable data cubing, reducing the number of cuboids by at least an order of magnitude or more in comparison with that of closed ones.

5.1.1 Motivation

Efficient computation of data cubes has been one of the focusing points in research since the introduction of data warehousing, OLAP, and the data cube operator [Gray et al. 96]. Data cube computation can be formulated as a process that takes a set of tuples as input, computes with or without some auxiliary data structure, and materializes the aggregated results for all cells in all cuboids. Its size is usually much larger than the input database, since a table with n dimensions results in 2^n cuboids. Thus, most work is dedicated to reduce either the computation time or the final cube size, such as efficient cube computation [Beyer & Ramakrishnan 99] [Shao et al. 04] [Han et al. 01] or cube compression [Lakshmanan et al. 02] [Xin et al. 06]. These cost reduction processes are all without loss of any information, while some others, like the approximation [Barbara et al. 97] or the iceberg-cube [Fang et al. 98] [Beyer & Ramakrishnan 99] [Xin et al. 06] [Han et al. 01] ones, reduce the costs by skipping trivial information.

The ideas of compressing cuboids cells in terms of classes of cells, or closed cells, seem to be an interesting approach to reduce size complexity, and also to explore optimally the semantics from the cube lattice. Nevertheless, for some of the dense databases we consider in this discussion, even the set of all closed cuboids cells would grow to be too large. The only recourse may be to mine the maximal cuboids cells in such domains. However, a pure maximal approaching implies loosing some information - one can generate the complete set of cuboids cells from its maximal but without their respective aggregation value. To play with some loss of information we propose a new measure, that we called the correlated value of a cuboid cell. This measure is inspired on ***all_confidence measure*** [Omiecinski 03], which has been successfully adopted for judging interesting patterns in association rule mining, and further exploited with confidence closed correlated pattern [Kim et al. 04]. This measure must disclose true correlation (also dependence) relationship among cuboids cells and needs to hold the null-invariance property. Furthermore, real world databases tend to be correlated, i.e., dimensions values are usually dependent on each other.

The main motivation of the proposed method emerged from the observation that real databases tend to be correlated, i.e., dimensions values are usually dependent on each other. For example, Store Wallgreens always sells Product Nappy or Store Starbucks always makes Product Coffee. In addition, the result of correlated cells on the corresponding data cube is to generate a large number of cells with same aggregation values. The Range CUBE method was the first approach to explore correlation among dimensions values by using a range trie [Feng et al. 04]. Although it does not compress the cube optimally and may not disclose true correlation relationship among cuboids cells holding the null-invariance property [Omiecinski 03] [Xiong et al. 03] [Kim et al. 04]. Inspired on the previous issues we raise a few questions to drive this work:

- *Can we develop an algorithm which captures maximal correlated cuboids cells on dense/sparse databases?*
- *How much such an approach can reduce the complete set of cuboids in comparison with the other approaches (i.e., pure maximal to closed ones)?*
- *How about the data cubing costs?*

In this section, we propose a new iceberg cube mining method for reducing cuboids aggregation ***by means of computing only the maximal-correlated cuboids cells***, and present the M3C-Cubing algorithm that brings out those cuboids.

5.1.2 Maximal-correlated Cuboid Cells

A cuboid is a multi-dimensional summarization of a subset of dimensions and contains a set of cells. A data cube can be viewed as a lattice of cuboids, which also integrates a set of cells. Next, we present few definitions to state the problem of finding maximal-correlated cuboids cells.

Definition 1 (***Cuboid Cell***) In an n-dimension data cube, a cell c = (i_1,i_2,…,i_n : m) (where m is a measure) is called a k-dimensional cuboid cell (i.e., a cell in a k-dimensional cuboid), if and only if there are exactly k ($k \leq n$) values among {i_1,i_2,…,i_n} which are not * (i.e., all). We further denote M(c) = m and V(c) = (i_1,i_2,…,i_n). In this paper, we assume that the measure m is count.

A cell is called iceberg cell if it satisfies a threshold constraint on the measure. For example, an iceberg constraint on measure count is M(c) ≥ min_supp (where min_supp is a user-given threshold). Given two cells $c = (i_1,i_2,\ldots,i_n : m)$ and $c' = (i'_1,i'_2,\ldots,i'_n : m')$, we denote $V(c) \leq V(c')$ if for each ij ($j = 1,\ldots,n$) which is not *, $i'_j = i_j$. A cell c is said to be covered by another cell c' if for each c'' such that $V(c) \leq V(c'') \leq V(c')$, $M(c'') = M(c')$. A cell is called a closed cuboid cell if it is not covered by any other cells. A cell is called a maximal cuboid cell if it is closed and has no other cell c which is superset of it (we have an exception just in case when its correlated value is higher than a minimum threshold).

Definition 2 (***The Correlated Value of a Cuboid Cell***) – Given a cell c, the correlated value 3CV of a V(c) is defined as,

$\mathrm{maxM}(c) = \max \{M(ci) | \text{for each } ci \in V(c)\}$ Eq.(1)

$3CV(c) = M(c) / \mathrm{maxM}(c)$ Eq.(2)

Definition 3 (***Maximal Correlated Cuboid Cell***) A cell c is a maximal-correlated cuboid cell (M3C) if it is covered by a maximal cuboid cell, its M(c) value is higher than min_supp and its 3CV(c) value is higher than min_3CV (where min_3CV is a user-given threshold for correlation).

From the last definition we allow a correlated exception for its supersets, where it is true when cell c is covered by another cell c' and 3CV(c') is higher than min_3CV. Given the above definitions, ***the problem of computing the maximal-correlated cuboids cells is to compute all maximal cuboids cells which satisfy iceberg constraints and its correlated exception cells***. An example of the maximal-correlated cuboids cells is given in Example 1.

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a1	b1	c1	d3
a1	b1	c1	d1
a2	b2	c2	d4

Table 16. Example of Maximal Correlated Cuboids Cells.

Example 1 - Maximal Correlated Cuboids Cells. Table 16 shows a table (with four attributes) in a relational database. Let the measure be count, the iceberg be count ≥ 2 and the correlated value $3CV \geq 0.85$. Then c1 = (a1,b1,c1,* : 3) and c2 = (a1,*,*,* : 4) are closed cells; c1 is a maximal cell; c3 = (a1,b1,*,* : 3) and c4 = (*,b1,c1,* : 3) are covered by c1; but c4 has a correlated exception (3CV=1); c5 = (a2,b2,c2,d4 : 1) does not satisfy the iceberg constraint. Therefore, c1 and c4 are maximal correlated cuboids' cells. The iceberg condition is *count ≥ min_sup* and *the correlated exception value ≥ min_3CV*.

5.1.3 M3C-Cubing

The proposed method for extraction of the Maximal-Correlated Cuboids Cells follows the BUC data cubing ideas [Beyer & Ramakrishnan 99] – we call it as M3C-Cubing. The computation starts from the smallest cuboids of the lattice, and works its way towards the larger, less aggregated cuboids. Our method does not share the computation of aggregates between parent and child cuboids, only the partitioning cost. Besides, as was verified by BUC experimental results, partitioning is the major expense, not the aggregation one.

We begin by defining the necessary terminology for describing the M3C-Cubing algorithm. We consider a base relation cell being a mapping K(c)→ M(c), where K is a composite key built from the grouping attributes values in V(c) and M(c) is also a composite key with the value to be aggregated. From the base relation cell we can extract several partitions; each partition has a subset of cells to aggregate. The partition of a base relation cell is defined as P(c)→ {K(c)→M(c)}, where P(c) is the partition key.

In order to get the correlated value of a cuboid cell (3CV) we need to keep the aggregation value for each 1-D cuboid. This is denoted as a mapping from 1-D(c)→ M(c). We should note that the maximum value will occur when the subset K(c) consists of a single grouping attribute (see Definition 2).

M3C-Cubing is guided by an SE-tree infrastructure, first introduced by Rymon [Rymon 92], and adopted later by Mafia [Burdick et al. 01] and Max-miner[Bayardo 98]. In this work, we call it as ***M3C-tree***. The M3C-tree is traversed by using a pure depth-first (DFS) order. Each node of the M3C-tree provides n-D cuboids which will be further partitioned, aggregated and checked if it is maximal or not (see Definition 1). In general, superset pruning works better with DFS order since many least aggregated cuboids may already have been discovered.

The strategies for pruning non-maximal correlated cuboids cells (nonM3C) basically attempt to: test out iceberg condition, check if it is maximal and when is not, check if it is a correlation exception (see Definition 3). They are just discarded in case its 3CV value is lower than a minimum threshold (min_3CV). Consequently, we provide the complete set of interesting cuboids which are maximal-correlated cuboids.

M3C-Cubing also keeps the current cuboids aggregated and the previous one for further pruning out of nonM3C cells. To speed up this process, we cannot remove an entire branch of the M3C-tree, since we have to aggregate its related partitions in order to validate the pruning conditions mentioned before. In this sense, we are just able to prune out nonM3C cells by the time we expand the M3C-tree level-by-level.

```
Pseudo M3C-Cubing Algorithm
Input:       a table relation trel; min supp; min 3CV.
Output      : the set of maximal-correlated cuboids cells.
Method      :
1. Let brel be the base relation of trel.
2. Build the M3C-tree concerning the grouping attributes in brel.
3. Call M3C-Cubing (min supp, min 3CV, brel, M3C-tree).

Procedure M3C-Cubing (min supp, min 3CV, brel, M3C-tree)
 1: get 1-D cuboids from M3C-tree
 2: for each j in 1-D cuboids do
 3:    get its partition from brel on dimensions [n], and
   set part ←{K(c)→M(c)}
 4:    aggregate part and set agg ← {V(c)→M(c)} when
M(c)>=min supp
 5:    set allCbs ← {agg}; set 3cv-1d ← {allCbs}
 6: end for
 7: get n-D cuboids in DFS order from M3C-tree
 8: for each k in n-D cuboids do
 9:    get its partition from brel on dimensions [n-D],
       and set part ←{K(c)→M(c)}
10:    aggregate part and set agg ← {V(c)→M(c)} when
M(c)>=min supp
11:    set allCbs ← allCbs  ∪ {agg}; set currCbs ← {agg}
12:    set 3cv-nd ← 3cv-nd  ∪ {call 3cv-nd(3cv-1d, agg )}
13:    set maxCbs ← maxCbs ∪
           {call maxCorr (allCbs, currCbs, 3cv-nd, min 3CV )}
14: end for
Procedure maxCorr(allCbs, currCbs, 3cv-nd, min 3CV)
  1: set nonM3C ←{ }
  2: for each j in currCbs do
  2:     for each k in allCbs do
  3:         if dom(allCbs) is superset of dom(currCbs),
              nonM3C ← nonM3C ∪ {dom(currCbs)}
  4:     end for
  5: end for
  6: remove any nonM3C in allCbs where 3cv-nd(nonM3C)< min 3CV
  7: return allCbs

Procedure 3cv-nd(3cv-1d, agg)
```

```
1: for each j in agg do
2:     splits into 1-D cells; maxValue=0;
3:     for each k in 1-D cells do
4:         get its aggValue(3cv-1d)
5:         if aggValue>=maxValue, maxValue=aggValue
6:     end for
7:     set 3cv=SI{agg}/ maxValue; set 3cv-nd ← {dom(agg) → 3cv}
8: end for; return 3cv-nd
```

Algorithm 5. The Pseudo-code of the Maximal-correlated cubing method.

With the aim of evaluating how M3C-Cubing reduces the final set of cuboids, we have to do a few modifications to the main method to support both pure maximal cuboids and closed ones. Those modifications are available as two new procedures: One for pure maximal and other for closed cuboids. We omit here those procedures, but one can also follows the definitions on section 5.1.2.

To bring out the pure maximal we just need to re-write the line 6 in *maxCorr* Procedure. Thus, the conditional test on *3CV* value of the ancestor cuboids is set apart. Needless to say, that we cannot make any use of *3CV-nd* procedure either to get pure maximal or closed cuboids. To get just the closed cuboids we must verify the *closedness* property [Xin et al. 06] among the cuboids, consequently we just have to check if its not covered by other cells.

Cover Equivalence in M3C-cubing

The idea of grouping cover partitions cells into classes can also be explored by M3C- Cubing in order to shrink even more the final data cube. By definition 1, it is possible to group a set of cuboids cells by verifying those cells which are cover equivalent ones [Lakshmanan et al. 02]. Thus, these cells essentially have the same value for any aggregate on any measure but with different degrees of correlation. For instance, in Example 1 the cells (a1,b1,c1,*) and (*,b1,c1,*) are cover equivalent cells. Next, we present a few concepts for guiding the grouping cover partition process with M3C-Cubing.

Cover partitions – The partition induced by cover equivalence is convex. Cover partitions can be grouped into a M3C class (ji..jn) (Figure 26). Each class in a cover partition has a unique maximal upper bound, and a unique lower bound (Table 17).

Upper bound cell – The upper bound for a particular class is the maximal cuboid cell contained in this class. Such as, in Example 1, the cell (a1,b1,c1,*) is the upper bound cell.

Lower bound cell – The lower bound cell for a particular class is the maximal 3CV value achieved by the correlated value of the cuboid cell (see Definition 2). Since M3C cubing allows catching all correlated exception cuboids, the lower bound cell will be that one with the highest 3CV value. E.g., in Example 1, the lower bound cell for class j1 is given by (*,b1,c1,*).

To explore maximal correlation over those classes we have to define the local_3CV value for each class. The local_3CV value of a class is the maximum local value given by the lower bound cell of each class (Table 17).

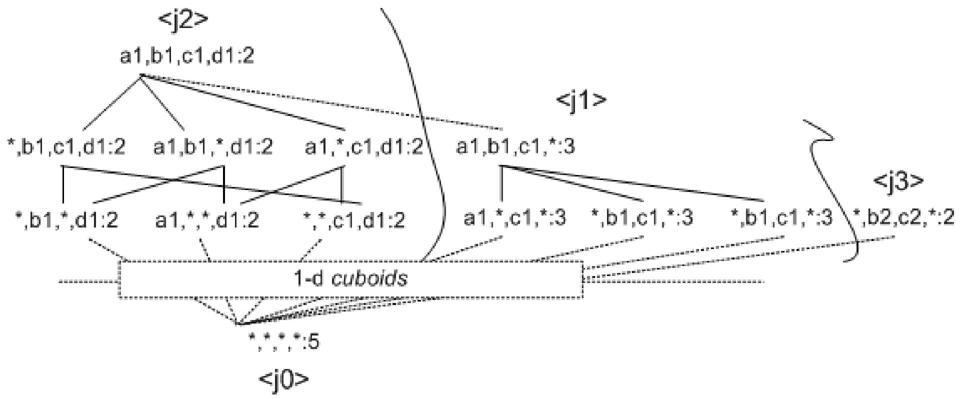


Figure 26. The M3C lattice formed by the cover partitions of the Example 1.

ClassID	UpBound	LoBound	Local_3CV	Lat_child	Agg
j0	(*,*,*,*)	(*,*,*,*)	0	-1	5
j1	(a1,b1,c1,*)	(*,b1,c1,*)	1	j0	3
j2	(a1,b1,c1,d1)	(*,b1,*,d1)	2/3	j0	2
j3	(*,b2,c2,*)	(*,b2,c2,*)	1	j0	2

Table 17. The set of classes from Example1.

5.1.4 Evaluation Study

In this section, we report our experimental results on compression and performance aspects of each method (M3C=Maximal-Correlated, Max = pure Maximal and Closed). The results are quite the same concerning to the performance point of view. This is true because those methods were developed having the basis on our main method (M3C). So, we can take those results only as an example of how much these modifications affect the whole time processing of the M3C-Cubing. On the other hand, reducing aspects of M3C-Cubing shows its viability by providing an interesting tradeoff between a pure maximal approaches to a closed one.

All the experiments were performed on a 3GHz Pentium IV with 1Gb of RAM memory, running Windows XP Professional. M3C-Cubing was coded with Java 1.5, and they were performed over eight synthetic datasets (Table 18). The values for columns *min_3CV%* and *min_supp%* are provided for the tests when fixing one value and varying the other. The density column shows the degree(%)[2] of density/sparseness of each dataset. All datasets have a normal distribution.

Dset	**Tuples**	**Dims**	**Card.**	**Density**	**Min_3CV%**	**Min_Supp%**
d1	100	3	3	27%	40%	2%
d2	250	5	3	97%	27%	0.80%
d3	500	3	5	25%	10%	0.60%
d4	750	4	5	83%	15%	0.67%
d5	1000	5	3	24%	17%	0.40%
d6	1250	4	6	100%	9%	0.16%
d7	3000	7	3	73%	35%	0.10%
d8	5000	6	4	82%	50%	0.04%

Table 18. The overall information of each dataset.

We first show that the complete set of maximal-correlated cuboids cells (M3C) is much smaller in comparison with both that of pure maximal (Max) and that of close ones (Closed). Figure 27 shows the number of cuboids generated by each approach from all datasets. The number of cuboids is plotted on a *log scale*. Figure 27(a) presents the number of cuboids generated when *min_3CV* is fixed and *min_supp* varies, while figure 27(b) shows those cuboids the other way around, fixing *min_supp* and varying *min_3CV*. These figures show that the M3C generates much smaller cuboids under lower *min_supp* or lower *min_3CV*. They also illustrate how much bigger the closed cuboids are in comparison with the other two methods. These results also indicate that under higher density datasets the chances of reducing cuboids by M3C-Cubing is more effective. Furthermore, the distance between Max and M3C reveals the gap of cuboids which are discarded (higher correlated ones) when using a pure maximal approach.

[2] The density degree of a dataset is calculated by the division: the product of each dimension (its cardinality value) by the number of tuples within the related dataset. The dataset is more dense when density degree is close to 100%.

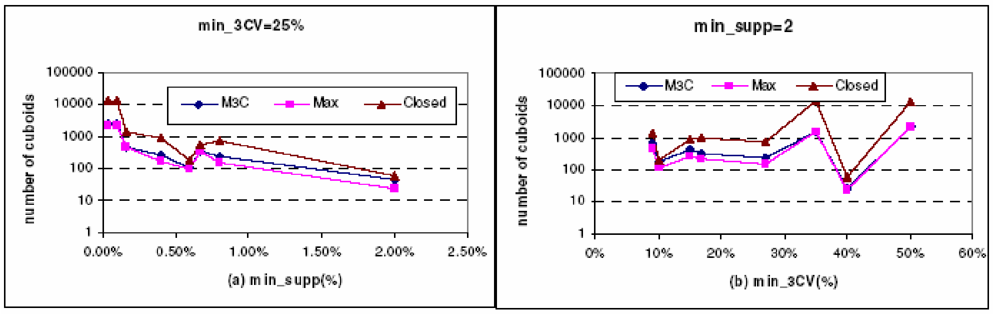


Figure 27. Number of cuboids generated from all datasets.

The next two figures (figure 28(a) and figure 28(b)) show the performance aspects of M3C-Cubing from all datasets. These figures follow the same configuration properties from previous two (figure 27(a) and figure 27(b)). Figure 28(a) illustrate that under a fixed *min_supp*, the maximal-correlated cuboids are useful only with lower *3CV* thresholds. This is confirmed by the *downward property* [12] of *3CV_value* of a cuboid cell. By the time the data cubing process is getting closer to the least aggregated cuboids, the *3CV_value* also decreases, so the computation time is pretty close, because *3CV* is decreasing. Figure 28(b) points out the effectiveness of M3C-Cubing under lower *min_supp*, giving more likelihood to identify correlated-cuboids, increasing a little-bit the processing time to prune out *nonM3C* cells.

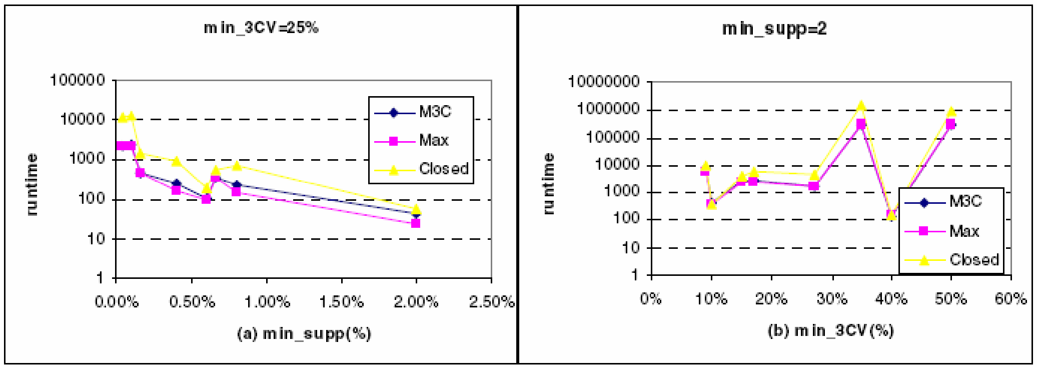


Figure 28. The execution time from all datasets.

Now, we are going to present a few examples concerning the reduction costs of computing the cube. Figures ranging from figure 29(a) to figure 29(d) illustrate those results. Figure 29(a) and figure 29(b) shows the number of cuboids generated when *min_supp* varies and *min_3CV* is fixed, while figures 29(c), 29(d) present those when *min_3CV* varies and *min_supp* is fixed. Under any circumstances the final cube is quite closer to the closed one, which points out that even using such closed-reduction the cuboids size remains even bigger. It is also demonstrates how much M3C-Cubing can save in comparison with the other methods.

In summary, the experimental results show that the number of maximal-correlated cuboids is quite smaller in comparison with that of the closed ones. Even, with a few modifications to the main features of M3C-Cubing, it still performs competitively with the other ones. Moreover, we provide the set of all maximal-correlated cuboids.

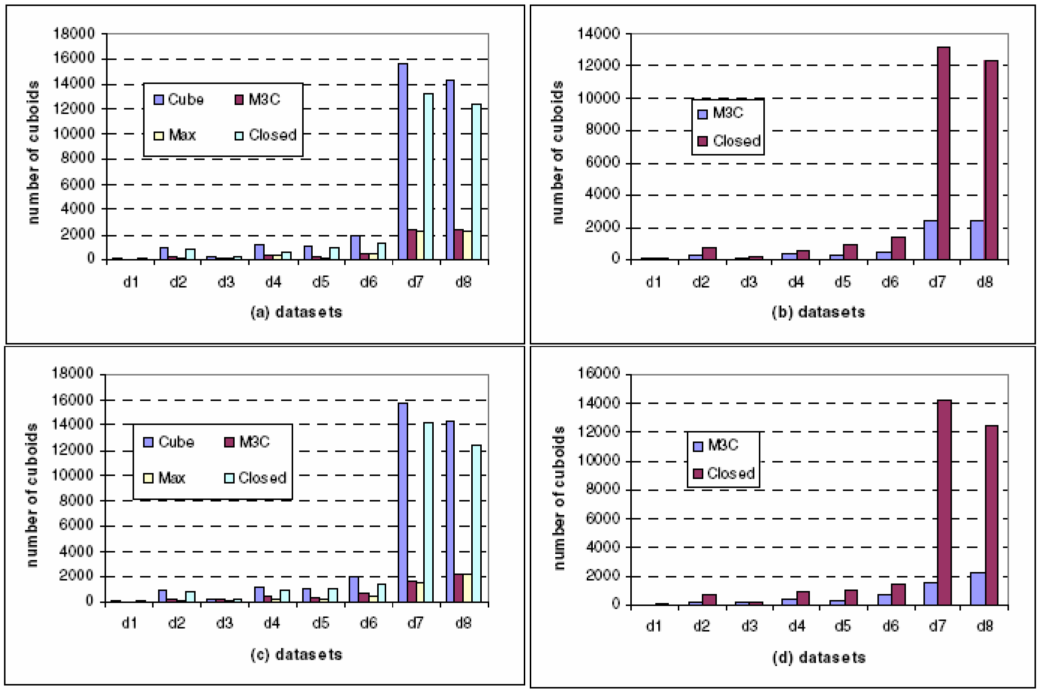


Figure 29. Reductions cost of cubing over all datasets.

5.1.5 Summary

The motivation behind iceberg cube mining is tightly related to reducing the search space for computing aggregates. The classical methods either offer ways for sharing the computation among cuboids or for exploring the partition costs in order to reduce the large output size.

We have presented ***M3C-Cubing that effectively reduces the complete set of cuboids cells introducing a new notion called maximal-correlated cuboids cells***. Through this cubing method we can find the maximal combinations among the grouping attributes and also keep its exceptional correlated cuboids, which can also indicates interesting changes on the cuboids during the cubing process.

We also have plans to investigate other aspects not addressed in this work such as: the application of 3CV measure over other aggregate functions (average, min, max…) and the issues related to recover an aggregation value of subcells of an M3C cell.

For efficient mining of those cuboids we have devised M3C-Cubing which is guided by an M3C-tree with a pure DFS traversal order. ***In order to improve pruning, we must investigate the tail information of each node in the M3C-tree such as designed in*** [Gouda & Zaki 05] [Zou et al. 02].

Finally, our evaluation study shows that maximal-correlated cuboids computation reduces the number of cuboids by at least an order of magnitude or more in comparison with the traditional approaches.

5.2 Aggregating Top-K Gradient Cells

Several business applications such as marketing basket analysis, clickstream analysis, fraud detection and churning migration analysis demand gradient data analysis. By employing gradient data analysis one is able to identify trends, outliers and answering what-if questions over large databases. Gradient queries were first introduced by [Imielinski et al. 02] as the cubegrade problem. The main idea is to detect interesting changes in a multidimensional space (MDS). Thus, changes in a set of measures (aggregates) are associated with changes in sector characteristics (dimensions).

MDS contains a huge number of cells which poses great challenge for mining gradient cells on a useful time. [Dong et al. 04] have proposed gradient constraints to smooth the computational costs involved in such queries. Even by using such constraints on large databases, the number of interesting cases to evaluate is still large.

In this section, we are interested to explore best cases (Top-K cells) of interesting multidimensional gradients. There several studies on Top-K queries, but preference queries with multidimensional selection were introduced quite recently by [Dong et al. 06]. Furthermore, traditional Top-K methods work well in presence of convex functions (gradients are non-convex ones). We have revisited iceberg cubing for complex measures, since it is the basis for mining gradient cells. We also propose a gradient-based cubing strategy to evaluate interesting gradient regions in MDS. Thus, the main challenge is to find maximum gradient regions (MGRs) that maximize the task of mining Top-K gradient cells. Our performance study indicates that our strategy is effective on finding the most interesting gradients in multidimensional space.

5.2.1 Gradient Queries

Gradient queries were first introduced by [Imielinski et al. 02] as the cubegrade problem. The main idea is to detect interesting changes in a multidimensional space (MDS). Thus, changes in a set of measures (aggregates) are associated with changes in sector characteristics (dimensions). By employing gradient data analysis one is able to identify trends, outliers and answering what-if questions over large databases. MDS contains a huge number of cells which poses great challenge for mining gradient cells on a useful time. To reduce search space, [Dong et al. 04] propose several constraints which are explored by a set-oriented processing. Even by using such constraints on large databases, the number of interesting cases to evaluate is still large. Furthermore, at a first experience for finding gradients in MDS, users often don't feel confident of what set of probe constraints to use. Thus, it should be the case to provide some Top-K facility over MDS.

As illustration of the motivation behind this problem, consider the following example of generating alarms for potential fraud situations on mobile telecommunications systems. Generally speaking, those alarms are generated when abnormal utilization of the system is detected, meaning that sensitive changes happened concerning the normal behavior. For example, one may want to see what are associated with significant changes of the average (w.r.t., the number of calls) in the Porto area on Monday compared against Tuesday, and the answer could be one in the form the average of number of calls for callings during working time in Campaña went up 55 percent, while those callings during night time in Bonfim went down 15 percent. Expressions such as callings during working time correspond to cells in data cubes and describe sectors of the business modeled by the data cube. Given that the number of calls generated in a business day in Porto area is extremely large (hundred million calls); the fraud analyst would be interest to evaluate just the Top-10 higher changes in that scenario, specially those ones in Campaña area. Then the fraud analyst would be able to drillthrough most interesting customers.

The problem of mining Top-K cells with multidimensional selection conditions and raking gradients is significantly more expressive than the classical Top-K and cubegrade query [Imielinski et al. 02]. In this section, we are interested to mine best cases (Top-K cells) of multidimensional gradients in a MDS.

5.2.2 Problem Formulation and Assumptions

A cuboid is a multidimensional summarization by means of aggregating functions over a subset of dimensions and contains a set of cells, called cuboid cells. A data cube can be viewed as a lattice of cuboids, which also integrates a set of cuboid cells. Data cubes are built from relational base tables (fact tables) from large data warehouses. Further we provide a set of definitions which allows us stating the problem of mining top-k gradient cells.

Definition 1 (***K-Dimensional Cuboid Cell***) In a n-dimension data cube, a cell c = $(i_1,i_2,\dots,i_n : m)$ (where m is a measure) is called a k-dimensional cuboid cell (i.e., a cell in a k-dimensional cuboid), if and only if there are exactly k ($k \leq n$) values among $\{i_1,i_2,\dots,i_n\}$ which are not * (i.e., all). If k = n, then c is a *base cell*. A base cell does not have any descendant. A cell c is an *aggregated cell* only when it is an ancestor of some base cell (i.e., where $k < n$). We further denote $M(c) = m$ and $V(c) = (i_1,i_2,\ldots,i_n)$.

Definition 2 (***Iceberg Cell***) A cuboid cell is called *iceberg cell* if it satisfies a threshold constraint on the measure. For example, an iceberg constraint on measure count is *M(c) ≥ min_supp* (where *min_supp* is an user-given threshold).

Definition 3 (***Closed and Maximal Cells***) Given two cells $c = (i_1,i_2,\ldots,i_n : m)$ and $c' = (i_1', i_2',\ldots,i_n' : m')$, we denote $V(c) \leq V(c')$ if for each i_j $(j = 1,\ldots,n)$ which is not *, $i_j' = i_j$. A cell c is said to be covered by another cell c′ if for each c″ such that $V(c) \leq V(c'') \leq V(c')$, $M(c'') = M(c')$. A cell is called a *closed cuboid cell* if it is not covered by any other cells. A cell is called a *maximal cuboid cell* if it is closed and has no other cell c which is superset of it.

Definition 4 (***Matchable Cells***) A cell c is said to match a cell c′ when they differ from each other in one, and only one, cube modification at time. These modifications could be: Cube *generalization/specialization* iff c is an *ancestor* of c′ and c′ a *descendant* of c; or *mutation* iff c is a sibling of c′, and vice versa.

Definition 5 (***Probe Cells***) Probe cells (pc) are particular cases of iceberg cells which are significant according to some selection condition. This selection condition is a query constraint on base table's dimensions.

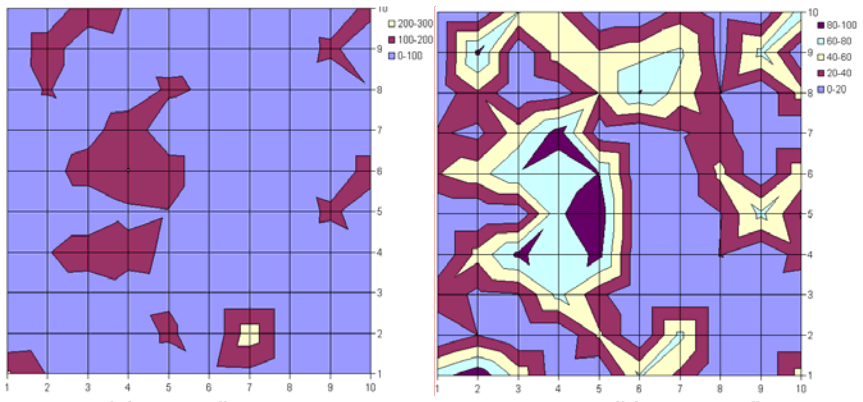
Definition 6 (***Gradient Cells***) A cell c_g is said to be gradient cell of a probe cell c_p, when they are matchable cells and their *delta change*, given by $\Delta g(c_g, c_p) \equiv (g(c_g, c_p) \geq \psi)$ is *true*, where ψ is a constant value and g is a *gradient function*.

In this section we confine our discussion with average gradients ($M(c_g)/M(c_p)$). Average gradients are effective functions for detecting interesting changes in multidimensional space, but they also pose great challenge to the cubing computation model [Dong et al. 04] [Han et al. 01].

Problem Definition. Given a base table *R*, iceberg condition *IC*, probe condition *PC*, the mining of *Top-k Multidimensional Gradients* from *R* is: Find the most interesting (Top_K) gradient-probe pairs (c_g, c_p) such that $\Delta g(c_g, c_p)$ is true.

5.2.3 Mining Top-K Gradient Cells

Before start the discussion about mining Top-K cells in MDS, lets first look to Figure 30. This figure shows how aggregating values are distributed along cube's dimensions. The base table used for cubing was generated according to a uniform distribution. It has 100 tuples, 3 dimensions (x, y, z) with cardinalities varying from 1 to 10, and a measure (m) varying from 1 to 100.



(a) *count()* (b) *average()*

Figure 30. It presents the distribution of aggregating values from a 2-D(x, y) cube.

From Figure 30 we can see that different aggregating functions will provide different density regions in data cubes. When searching for gradients, one may want to start this task by evaluating a region in which presents higher variance. In Figure 30(b) the central region corresponding to the rectangle *R1 {[4, 7] : [6, 5]}* covers all five *bins {b_0={0-20},b_1={20-40}, b_2={40-60}, b_3={60-80}, b_4={80-100}}*. If it is possible to identify such region, chances are that the most interesting gradients will take place there. We denote those regions as *gradient regions (GR)*. The problem is that average is an algebraic function and it has a *spreading factor (SF)* with respect to the distribution of aggregating values over the cube quite unpredictable. Thus, there will be sets of possible gradient regions to looking for in the cube before selecting the most Top-K gradient cells.
Another interesting insight from Figure 30(b) is that gradients are maximized when walking from bins b_o to b_4. Therefore, even if a region doesn't cover all bins but if at least has the lowest and highest ones; it should be a good candidate *GR* for mining Top-K cells. We expect that *GRs* with largest aggregating values will provide higher gradient cells. This observation motivates us to evaluate gradients cells by using a ***gradient ascent approach*** [Navidi 2007].

Gradient ascent is based on the observation that if the real-valued function *f*(x) is defined and differentiable in a neighborhood of a point *Y*, *f*(x) then increases fastest if one goes from *Y* in the direction of the gradient of *f* at *Y*, Δ*f*(Y) . It follows that, if

$$Z = Y + \gamma \Delta f(Y)$$

Equation 16. The gradient ascent equation, (Part I).

for, γ > 0 a small enough number, then *f*(Y) ≤ *f*(Z). With this observation, one could start with x_0 for a local maximum of *f*, and considers the sequence $x_0, x_1, x_2, \dots$ such that

$$x_{n+1} = x_n + \gamma_n \Delta f(x_n), n \geq 0$$

Equation 17. The gradient ascent equation, (Part II).

We have *f*(x0) ≤ *f*(x1) ≤ *f*(x2) ≤ …, so we expect the sequence (x_n) converge to the local maximum. Therefore, when evaluating a *GR* we first search for the *maximal probe cells*, i.e. the highest aggregating values (non-closed and non-maximal cells) on it and then calculating its gradients from all possible *matchable cells*.

To allow this gradient ascent traversal through cube's lattice, one needs to incorporate the main observations mentioned above into the cubing process.

Spreading Factor

The spreading factor is measured by the variance. This statistical measure indicates how the values are spread around the expected value. From 1-D cuboid cell we want to capture how large the differences are in a list of measures values ML=(M1,…,Mn) from the base relation R. Thus, dimensions are projected onto cube's lattice according to this variation. This variation follows

$$\frac{\sum (X - \mu)^2}{N}$$

, X is a particular value from ML, μ is the mean from ML and N is the number of elements in ML.

Equation 18. The spreading factor equation.

For large datasets one could use a variation of Equation 18, using a sample rather than the population. In this sense one can replace the denominator N by N-1.

Aggregating Gradient Cells

Our cubing method follows Frag-Cubing approach [Li et al. 04]. We start by building inverted indices and value-list indices from the base relation R, and then assembling high-dimensional cubes from low-dimensional ones. For example, to compute the cuboid cell {x1, y3, *} from Table 19, we intersect the tids (see Table 20) of x1 {1, 4} and y3 {4} to get the new list of {4}.

tid	**X**	**Y**	**Z**	**M**
1	x1	y2	z3	1
2	x2	y1	z2	2
3	x3	y1	Z2	3
4	x1	y3	z1	4

Table 19. Example of a base table relation R.

The order in which we compute all cuboids is given by a processing tree GR_{tree} following a DFS traversal order. Before creating GR_{tree} we can make use of the first heuristics for pruning (p_1, **pruning non-valid tuples**). To smooth the evaluation of iceberg-cells, it is interesting to build such tree only with tuples satisfying the iceberg condition.

After applying p_1, we are able to calculate the spreading factor (*sf*, see Section 5.2.3.1) of all individual dimensions X, Y and Z. From Table 20 we may say that we have a total of nine GRs to grow.

Attr.Value	**Tid.Lst**	**Tid.Sz**	**sf**
x1	1, 4	2	2.25
x2	2	1	0
x3	3	1	0
y1	2, 3	2	0.25
y2	1	1	0
y3	4	1	0
z1	4	1	0
z2	2, 3	2	0.25
z3	1	1	0

Table 20. Inverted indices and spreading factors for all dimensions of Table 19.

The GR_{tree} will follow the order of X>>Y>>Z. Given that we want to find large gradients, chances are that higher gradients will take place on projecting GRs having higher spreading factors. Here comes the second heuristic for pruning (p_2, ***pruning non-valid regions***). For example, a threshold constraint on a GR is *sf(GR) ≥ min_sf* (where *min_sf* is a user-given threshold). Let's say that we are interested to search for gradients on GRs having a sf ≥ 0.25. From this constraint we only need to evaluate 3 GRs rather than 9 ones. Cubing is carried out through projecting x_1; y_1 and finally z_2 (see Figure 31).

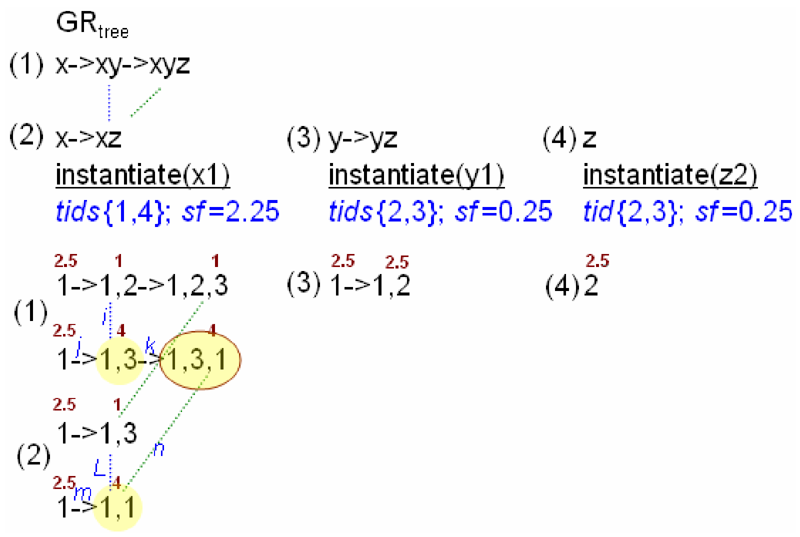


Figure 31. The lattice formed by projecting GRs {x1, y1, z2}. Aggregating values are placed upper all cuboid cells. Possible probe cells are denoted with shadow circles. The letters matchable cells shows valid gradient pairs.

Since we have cubing those three regions in Figure 31, it is possible now to mine the Top-K gradient cells from them. After this cubing, we also augment each region with its respective *bin* [agg_{min}; agg_{max}] according to the minimum and maximum aggregating value on it. For example b_{x1}=[1; 4], b_{y1}=[2,5; 2,5] and b_{z2}=[2,5; 2,5]. With all those information we can make use of the third heuristic for pruning (p_3, ***pruning non-TOPk regions***). Given that we have an iceberg condition on average such as $M_{avg}(c) \geq 2.7$, we can confine our search for Top-K cells only for GR=x_1.

Even by using such constraint, the number of gradient pairs to evaluate is still large in x_1. So, we must define our set of probe cells pc{} to mine. Remember that discussion about gradient ascent (section 5.2.3); by taking the local maximum (i.e., a cuboid cell having the highest aggregating value) from a particular region, all matchable cells containing this local maximum will increase gradient factors. Thus, our probe cells are given by the set of the Top-K aggregating values in that region $TopK_{pc}$. For a cuboid cell to be eligible as a probe cell, it cannot be a *closed and maximal cell*. For example, in Figure 31 the cuboid cell = {1,3,1} cannot be selected as a probe cell. Next, for each probe cell in $TopK_{pc}$ we calculate its gradients. Finally we select the Top-K gradient cells $TopK_{gr}$. For example, with we are looking for the Top-3 cell, the $TopK_{pc}$ is formed by {{1,3};{1,1};{1}}and $TopK_{gr}$ is formed by letters {i, L, j}.

Usually we will have more valid-ToPk regions rather than in the previous example. Thus, the final solution must take into account all local $TopK_{gr}$ before raking TOP-K cells. Besides, after having calculated all local $TopK_{gr}$, we continue searching for other valid gradient cells resulting from matchable cells (i.e., projecting probe cells from a GR_i over cuboid cells in GR_j).

```
Pseudo TopKgr-Cube Algorithm
Input:      a table relation trel; an iceberg condition on average
minavg; an iceberg condition on GR min sf; the number of Top-K
cells K
Output      : the set of gradient-cell pairs TopKgr.
Method      :
1. Let ftrel be the relation fact table
2. Build the processing tree GRtree // evaluating heuristic p1
3. Call TopKgr-Cube ().

Procedure TopKgr-Cube (ftrel, GRtree, minavg, min sf, K)
1: Get 1-D cuboids from GRtree
2: For each 1-D cuboids do {
3:  Set GR ← {subTree_GR(V(c))} //build gradient regions
4: For each GR having sf • min sf do //apply p2
  Aggregate valid GR // do projections, cubing
5: For each valid GR having a bin[min,max] satisfying minavg //
apply p3
6:  Set TopKpc ← {topKpc_GR(GR,K)} //select probe cells
7: For each valid TopK GR
    Set TopKgr ←{topKgr_GR(TopKpc)} // calculate its gradients
8: Rank Top-K cells (TopKgr,K) //rank gradient cells,
DESC order of ψ
```

Algorithm 6. Pseudo-code of the top-k gradient cubing method.

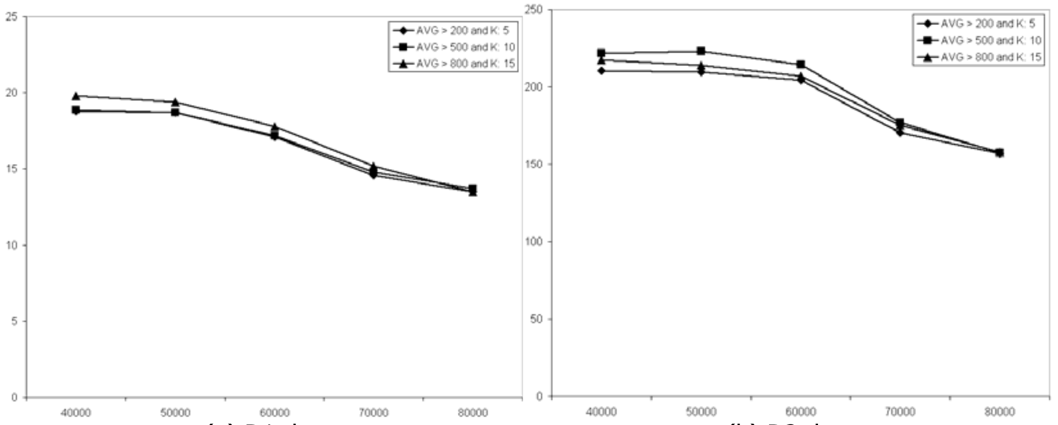
5.2.4 Evaluation Study

All the experiments were performed on a 3GHz Pentium IV with 1Gb of RAM memory, running Windows XP Professional. TopKgr-Cube was coded with Java 1.5, and it was performed over synthetic datasets (Table 21). These datasets were generated using the data generator described in [Dong et al. 03] [Han et al. 01].

Dset	Tuples	Dims	Card*	M[min,max]
D1	5000	7	100	100,1000
D2	10000	10	100	100,1000

Table 21. The overall information of each dataset.

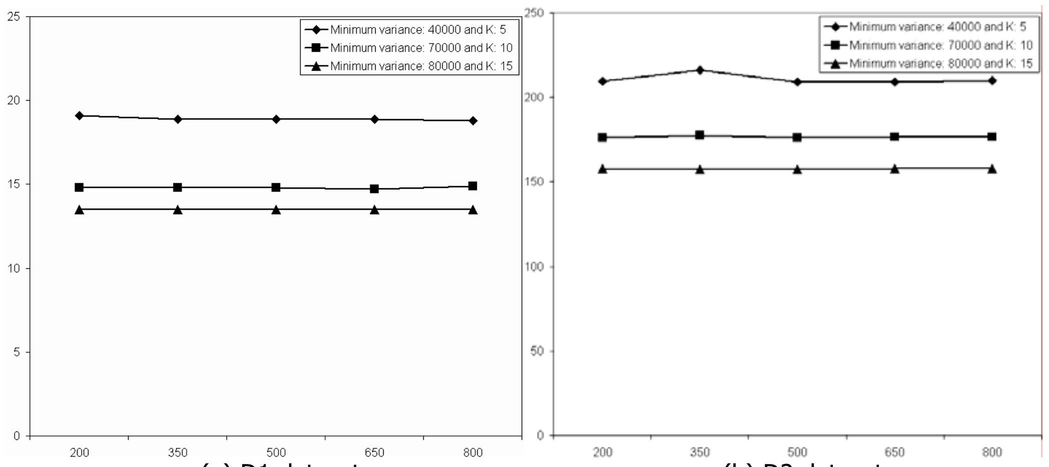
When running the next performance figures, we confine our tests with TopK$_{gr}$-Cube to the following scenarios: *one*, (Figure 32) we want to see the effects of using variance as a criteria for mining Top-K cells: *two*, (Figure 33) we want to see the effects of iceberg condition while selecting interesting regions; and *three*, (Figure 34) the pruning effects by using the heuristics used by our Top-K cubing method.



(a) D1 dataset (b) D2 dataset

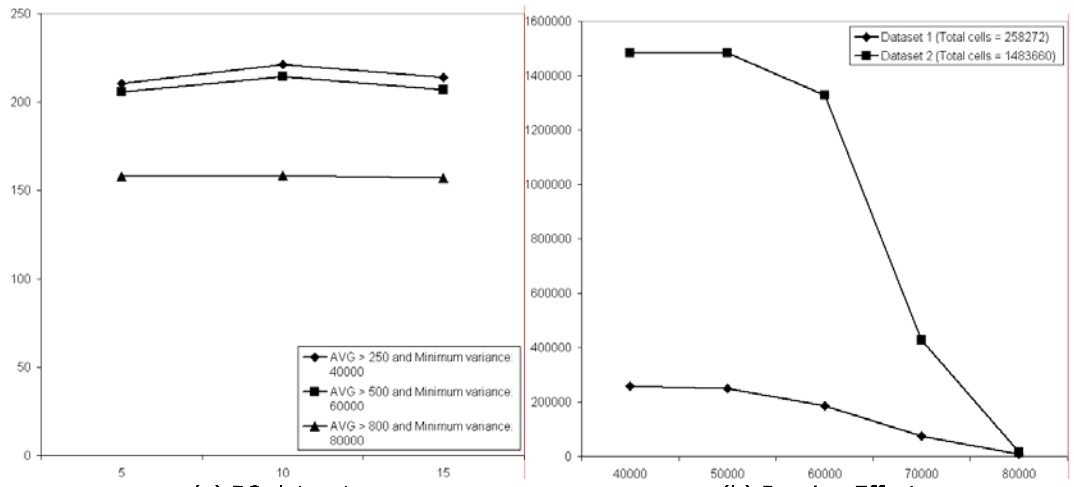
Figure 32. Performance (Runn.time(s), Y-axis) x Min_sf (X-axis).

* Those cardinalities provide very sparse data cubes, thus, poses more challenge to the Top-K cubing computation model.



(a) D1 dataset (b) D2 dataset

Figure 33. Performance (Runn.time(s), Y-axis) x Min_AVG (X-axis).



(a) D2 dataset (b) Pruning Effects

Figure 34. (a) Performance x K effects (X-axis). (b) Valid-cells (Y-axis) x Min_sf (X-axis).

Related Work

The problem of mining changes of sophisticated measures in a multidimensional space was first introduced by [Imielinski et al. 02] as a cubegrade problem. The main idea is to explore how changes (delta changes) in a set of measures (aggregates) of interest are associated with changes in the underlying characteristics of sectors (dimensions). In [Dong et al. 04] was proposed a method called LiveSet-Driven leading to a more efficient solution for mining gradient cells. This is achieved by group processing of live probe cells and pruning of search space by pushing several constraints deeply. There are also other studies [Sarawagi et al. 98a] [Sarawagi & Sathe 00] [Sathe & Sarawagi 01] for mining interesting cells on data cubes. The idea of interestingness in these works is quite different from that explored by gradient-based ones. Instead of using a specified gradient threshold in relevance to the cells' ancestor, descendants, and siblings, it relies on the statistical analysis of neighborhood values of a cell to determine its interestingness (or also outlyingness).

Those previous methods employ the idea of interestingness supported by either statistical or ratio-based approach. Such approach still provides a large number of interesting cases to evaluate, and on a real application scenario, one could be interested to explore just such a small number (best cases of Top-K cells) of gradient cells. There are several research papers on answering Top-queries [Chang et al. 00] [Hristidis et al. 01] [Bruno et al. 02] on large databases, which could be used as baseline for mining Top-K gradient cells. However, the range of complex delta functions provided by the cube gradient model complicates the direct application of those traditional Top-K query methods.

To the best of our knowledge, the problem of mining Top-K gradient cells in large databases is not well addressed yet. Even the idea of Top-K queries with multidimensional selection was introduced quite recently by [Dong et al. 06]. Furthermore, the model still relies on the computation of convex functions.

Conclusions

In this section, we have studied issues and mechanisms on effective mining of Top-K multidimensional gradients. Gradient are interesting changes in a set of measures (aggregates) associated with the changes in the core characteristics of cube cells. We have also proposed a gradient-based cubing strategy (TopKgr-Cube) to evaluate interesting gradient regions in MDS. This strategy relies on cubing gradient regions which show high variance. Thus, the main challenge is to find maximum gradient regions (MGRs) that maximize the task of mining Top-K gradient cells through a set of Top-K probe cells. To do so, we use a gradient ascent approach with a set of pruning heuristics guided by a specific GRtree.

Our performance study indicates that our strategy is effective on mining the most interesting gradients in multidimensional space. To end up our discussion we set the follow topics as interesting issues for future research:

- ***Top-K average pruning into GRs***. Although, we make pruning of GRs according to an iceberg condition. One can take advantage of Top-K average [Han et al. 01] for cubing only GRs satisfying this condition.
- ***Looking ahead for Top-K gradients***. Given that GR1>>GR2>>GR3, it should be the case that by looking ahead for a gradient cell in GR2 will not generate gradients higher than that in GR1.
- ***Mining high-dimensional Top-K cells***. The idea is to select small-fragments [Li et al. 04] (with some measure of interest) and then explore on-line query computation to mine high-dimensional Top-K cells.

Chapter 6

Conclusions and Future Work

6.1 Summary

In many applications, data contains structured information that is multidimensional and multilevel in nature, such as that ones in areas like e-commerce, telecommunications, retail, stocks, scientific-data, etc. Since last decade, we have been facing several research efforts on Data Warehousing and OLAP to get better view of multidimensional data, allowing a data-driven search for interesting patterns at several levels of data abstraction. This strategy of searching patterns on multidimensional databases is also called OLAPing (or data cubing). The basis of MDA relies on effective and efficient computation of aggregating functions.

MDA on higher dimensional data is a quite hard task; given the limitation of the most known cubing algorithms. So, *how to enhance this data-driven search with discovery-driven features smoothing the curse-of-dimensionality problem?* Besides, data as well as patterns evolve over time-to-time. Thus, *how to highlight those significant changes? Multidimensional Data Mining* (MDM) take its place helping to handle those previous issues. In summary, MDM attempts to combine ideas of cubing and mining techniques to get better mechanisms for multidimensional data analysis.

Clearly, MDA requires suitable multidimensional data mining (MDM) methods. It is not an easy task to decide what level of aggregation-based mining is necessary for revealing interesting spots in DM. In addition, it poses new challenges for novel uses of data cubing and data mining technology. Therefore, the work presented in this thesis lies outside of traditional work on OLAP mining. These previous discussions motivated us to devise several methods to achieve MDM through aggregation-based mining strategies.

These methods cover several levels of multidimensional data aggregation, from single-dimensional to n-dimensional, as well as, mining of interesting patterns. Apart from the methods, we have also designed new measures of interestingness for guiding MDA through high dimensional databases.

Three major issues were addressed to support the central thesis statement of this research, as follow:

- It is possible ***to achieve a simple infrastructure for MDM***, i.e., by combining traditional frequent- and sequential-pattern mining methods.
- It is possible ***to evaluate evolving multidimensional and multilevel patterns*** through providing different levels of aggregation-based mining methods, i.e., from single- to n-dimensional mining.
- It is possible ***to achieve advanced MDA*** by pushing and evaluating measures of interestingness during the processing step, i.e., cubing-based mining of high dimensional databases.

6.2 Contributions of this Thesis

We now summarize the main contributions accomplished in this research. These contributions are parts of several aggregation-based mining strategies provided in this thesis:

- ***A frequent pattern mining strategy closer to RBDMS***. Most of the itemset mining approaches are memory-like and run outside of the database. On the other hand, when we deal with data warehouse the size of tables is extremely huge for memory copy. We devised a pattern growth mining approach by means of database programming for finding all frequent itemsets. The main idea is to avoid one-at-a-time record retrieval from the database, saving both the copying and process context switching, expensive joins, and table reconstruction. This method is described in Chapter 2 and had been published in [Alves & Belo 05b].
- ***A hybrid method for mining inter-dimensional patterns***. Classical association rules are by nature intra-transaction based, i.e., the associations occurs among the

items within the same transaction. Further, those transactions are usually associated with a dimensional context which is typically ignored. This prevents that patterns, occurring along this dimension (i.e. associated context), are discovered. We devised a strategy for the extraction and analysis of inter-dimensional patterns. We also have explored a measure called cvd which allow us to reject patterns that presents significant distance variation, and thus, avoiding the representation of misleading patterns. This method is described in Chapter 2 and had been published in [Ferreira el al. 05].

- ***A family of aggregation-based single-dimensional mining methods***. We studied the problem of superimposed fraud detection in telecommunication systems. We proposed three aggregation-based single-dimensional mining methods for anomaly detection. These methods rely on a single level of aggregation called signatures. The first method is a signature deviation-based approach while the second one is designed for dynamic clustering analysis. Furthermore, we enhanced those two methods by using a graph-based strategy representing the social or community behaviour, without losing information, and also providing a rich structure for revealing abnormalities. We used annotated digraphs to represent the call graph network. The dynamics of changes are modeled into the graphs by weighting factors using damping ideas. New information is aggregated into this call graph in accordance with this representation and evaluated through pre-defined recursive functions. All these methods are described in Chapter 3 and were published in [Alves et al. 06a] [Ferreira et al. 06].
- ***An incremental cube-based mining method for mining multidimensional and multilevel patterns***. The updating processes on real DM applications implies that, the summarization made by cubing methods need to handle the new information available with incremental and discovery facilities. We described a method which is quite competitive w.r.t. other known cubing strategies, also providing an effective cube-based mining strategy for discovering knowledge over evolving DM. We also demonstrate that by exploring correlated cuboids during the discovery process we are able to provide an essential tradeoff between processing time and pattern accuracy. Such strategy is presented in Chapter 4 and had been published in [Alves & Belo 07a]
- ***Advancing aggregation-based mining methods***. Addressing advancing aggregation-based mining in high dimensional databases requires different kinds of pushing constraints into the aggregation process, since high MDA suffers from the

curse of dimensionality problem. We propose two distinct methods for MDA in high dimensional spaces. The first strategy is a promising candidate for scalable data cubing, reducing the number of cuboids by at least an order of magnitude or more in comparison with that of closed ones. This is achieved by pushing and evaluating correlated cuboids during processing time. The second method attempts to find interesting changes (Top-K cells) in high n-dimensional spaces. We reviewed iceberg cubing for complex measures, since it is the basis for mining gradient cells. We also propose a gradient-based cubing strategy to evaluate interesting gradient regions in MDS. Thus, the main challenge is to find maximum gradient regions (MGRs) that maximize the task of mining Top-K gradient cells. Both advanced aggregation-based mining methods are described in Chapter 5 and were published in [Alves & Belo 06b] [Alves & Belo 07b].

6.3 Future Work

Several directions can be exploited as a continuation of this research. We discuss and highlight a few technical challenges in this direction according to each topic discussed in this thesis.

6.3.1 Mining Intra- and Inter-dimensional Patterns

Frequent pattern mining (FIMI) is one of the basic tasks in data mining. Furthermore, it has been improved over the last decades through several research studies. Numerous methods have been developed to support the extraction of different kind of patterns. We have proposed two main strategies to discovery intra and inter-dimensional patterns. By combining classical FIMI methods we were able to provide a simple but effective strategy for mining multidimensional patterns. As a continuation of this work we suggest the next issues for further research:

- ***With respect to mining intra-dimensional patterns***. One issue that is controversial is code portability, in sense that PGS is tightly dependent of the database programming language with SQL-Extensions. On the other hand for huge databases, one might be interested to take all the advantages offered by the RDBMS. As future research, one should work on method enhancements for making this strategy more interactive, constrained and incremental. Furthermore, some work on improving the whole performance of PGS by using Table Variables (TV), which allows mimicking an array, instead of using database cursors, it is also very interesting topic for further research with PGS.

- ***With respect to mining inter-dimensional patterns***. So far, the cvd filter had been used as post-processing step of the sequence mining step. Although the performance of this strategy does not imply a bottleneck figure, it can be further improved. An interesting issued for future study is to push directly this pruning mechanism in the mining process. This is non trivial task, since this measure is neither monotonic nor anti-monotonic. Ideally, one should be concerned to devise a unique support measure, i.e. a measures combining both frequent and sequence threshold for mining inter-dimensional patterns.

6.3.2 Agregation-based single-dimensional Mining

Traditional count-based mining strategies such as FIMI ones relies on simple aggregating functions. We have proposed single-dimensional mining methods employing more complex functions which were properly evaluated on a real application scenario. We smooth high-dimensional computation by electing one dimension to profile at a time, and finally enhancing this single-dimensional model with several classical mining methods. Although, those methods were deeply investigate on a Telecommunication problem. We suggest the following issues as continuation of this study:

- ***With respect to aggregation-based deviation mining***. The most sensitive part of this approach relies on the evaluation step of the distance among signatures (section 3.2.3). Different distance functions may have different weighting factors. This allows us to detect several possible deviation scenarios, i.e., higher and low variability of the features variables involved in this evaluation. The main concern in the current approach relies on the manual settings of those weighting factors by fraud analysts. An interesting study would be to use some supervised/ weighting algorithm for suggesting analyst to use some particular setting which had been used successfully on similar running scenario in the past. In fact, the importance of feature variables used at each running scenario should take into account in this kind of evaluation. Finally, the maintenance of tables and indexes should be explored with more experimental studies in order to avoid the problem of rebuilding DB structures while sliding the window frame from w to w+1.

- ***With respect to aggregation-based clustering mining***. The proposed clustering approach is more or less the k-means/k-medoid approach, and has some issues when clustering very high-dim. Several other clustering methods should be explored here to alleviate the curse-of-dimensionality problem. Furthermore, since we compute an N*N matrix of distance measures of signatures, some feature selection method could be applied to point out important signatures for further analysis. On the other hand one should be concerned to define the importance criteria of a signature in this kind of application. On a daily basis scenarios with huge data stream it is not a trivial task applying clustering algorithms on N*N matrices. A comparison analysis should be made with other dynamic clustering algorithms to evaluate real gains of our chunk-based partitional clustering against other existent in the referenced literature.
- ***With respect to aggregation-based graph mining***. The score function we devised for detecting interesting call graph patterns takes into account usage patterns from a calling. In this sense, other graph component metrics should take place in order to incorporate measures of interesting concerning social impacts of that particular user/call in the whole call graph network. For instance, the work presented in [Nanavati et al. 06] employ several metrics such as pageRank, clustering coefficients, strong and weak components, among others for detecting significant graph patterns. On the other hand, in this study the authors do not consider the usage aspect when dealing with weighting (call co-occurrences) direct graphs. We believe that our proposed strategy will be enhanced by adding more score graph functions. Next, one could make use of some order statistics method to filter out the most (Top-K) significant call graph patterns. In this particular discussion, important nodes and communities from the whole call graph network. Given the huge data size associated to this kind of application, one should clearly take good benefits from using a Top-K approach to highlight interesting calls for further analysis.

6.3.3 Aggregation-based n-dimensional Mining

To allow aggregation-based mining on multidimensional space, we proposed a cube-mining strategy that can handle data cubing with several hierarchies, as well as, extraction and evaluation of multilevel and multidimensional patterns. Furthermore, it is also an incremental procedure. The following aspects would be interesting roads for future research on mining evolving data cubes:

- ***With respect to aggegation with several hierarchies***. Dimension hierarchies are actually a very complex area. They usually cannot be computed by producing new tables for each distinct hierarchy combination even by using arrays as proposed here. Although, we presented a method which takes this naïve implementation this problem is alleviated by employing measures of interestingness such as described on chapter 5. In practice, it has been shown that this can produce hundreds of thousands of views or more in real DM. Thus, future work must take into account other approaches like Dwarf cube [Sismanis et al. 02] and the CURE cube [Morfonios & Ioannidis 06] as comparison basis when cubing with several hierarchies.
- ***With respect to aggregation-based mining***. In terms of the mining work, we made use of Apriori-based approach for rule mining. It should be interesting to evaluate other rule mining method such as the FP-Growth (FP) [Han et al 00]. This method is very robust for mining large patterns. One should take as example the PGS strategy [Alves & Belo 05b] described in chapter 2, as an initial step for developing a similar method for rule mining with high dimensional DM. Even though, the frequent pattern mining discussed in the chapter 4 was not pure Apriori-like and indeed cube-mining, one should benefit from using a FP strategy for discovery interesting relations on DM. Finally, we suggest more experimental figures concerning cubing such as: a) use Zipf to control the skew of the data, and vary Zipf from 0 to 3 (0 being uniform); b) cubing process supports only distributive aggregate functions: sum(), maximum(), minimum(), and count(), future work must incorporate complex measures such as average(), std(), mean(). Take as an example the method discussed in [Alves et al. 07b].

6.3.4 Advanced Aggregation-based Mining

We have presented two methods for advanced aggregation-based mining by exploring measures of interestingness while cubing. We have also achieved a fair tradeoff between cube computation, compression and pattern discovery in a multidimensional space. To improve high-dimensional data analysis, taking as basis those proposed methods; we enumerate the following future works:

- ***With respect to mining correlated-cuboid cells***. We suggest evaluating the application of 3CV measure over more complex aggregating functions such as average, min, max, among others. In addition, one should also be concerned to recovering issues in M-D space, i.e., generating an aggregation value of subcells of an M3C cuboid cells. For efficient mining of those cuboids we have devised M3C-Cubing which is guided by an M3C-tree with a pure DFS traversal order. In order to improve even more the pruning step, one must investigate the tail information of each node in the M3C-tree such as designed in [Gouda & Zaki 05] [Zou et al. 02].

- ***With respect to mining Top-K gradients***. We can enumerate the following interesting paths to be explored: a) *Top-K average pruning* into GRs. Although we make pruning of GRs according to an iceberg condition, one can take advantage of Top-K average [Han et al. 01] for cubing only GRs satisfying this condition; b) *Looking ahead for Top-K gradients*. Given that GR1>>GR2>>GR3, it should be the case that by looking ahead for a gradient cell in GR2 will not generate gradients higher than that in GR1; c) *Mining high-dimensional Top-K cells*. The idea is to select small-fragments [Li et al. 04] (with some measure of interest) and then explore on-line query computation to mine high-dimensional Top-K cells.

The research efforts discussed so far in this thesis contributes with a set of methods for making practical and effective *multidimensional data analysis* (MDA) through multidimensional aggregation-based mining strategies. All these methods handle MDA from different perspectives. Thus, they are quite dependent on the kind of analysis and the level of dimensions to explore. To conclude this thesis, based on the review process and discussions at all conference meetings we attended, we can select our Mining Top-K Gradient Method as a promising one for further research and application on high-dimensional data analysis such as the ones required in biological databases.

Bibliography

[Agrawal & Shim 96] Agrawal, R., Shim., R.: Developing tightly-coupled data mining application on a relational database system. In Proc. of the 2nd Int. Conf. on Knowledge Discovery in Database and Data Mining, Portland, Oregon (1996).

[Agrawal & Srikant 94] Agrawal, R., Srikant., R.: Fast algorithms for mining association rules. In Proc. of the 20th Very Large Data Base Conference (1994) 487–499.

[Agrawal & Srikant 95] Agrawal, R. and Srikant, R.: 1995. Mining Sequential Patterns. In *Proceedings of the Eleventh international Conference on Data Engineering* (March 06 - 10, 1995). P. S. Yu and A. L. Chen, Eds. ICDE. IEEE Computer Society, Washington, DC, 3-14.

[Agrawal et al. 96] Agrawal, R., Mannila, H., Srikant, K., Toiven, H., Verkano, A.: Fast Discovery of Association Rules. Advances in Knowledge Discovery and Data Mining 1996: 307-328.

[Agrawal et al. 93] Agrawal, R., Imielinski, T., Swami, A..: Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Intl. Conference on Management of Data (1993) 207–216.

[Ale & Rossi 00] Ale, J. M. and Rossi, G. H.: 2000. An approach to discovering temporal association rules. In *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1* (Como, Italy). J. Carroll, E. Damiani, H. Haddad, and D. Oppenheim, Eds. SAC '00. ACM Press, New York, NY, 294-300. DOI= http://doi.acm.org/10.1145/335603.335770

[Alves & Belo 05a] **Alves, R.**, Belo, O.: Integrating Pattern Growth Mining on SQL-Server RDBMS. Technical Report-003, University of Minho, Department of Informatics, May (2005) http://alfa.di.uminho.pt/~ronnie/files_files/rt/2005-RT3-Ronnie.pdf

[Alves & Belo 05b] **Alves, R.**, Belo, O.: Programming Relational Databases for Itemset Mining over Large Transactional Tables. *EPIA 2005*: 314-324. DOI= 10.1007/11595014_32.

[Alves & Belo 06b] **Alves, R.**, Belo, O.: On the Computation of Maximal-Correlated Cuboids Cells. *DaWaK 2006*: 165-174. DOI = 10.1007/11823728_16.

[Alves et al. 06a] **Alves, R.**, Ferreira, P., Belo, O., Lopes, J., Ribeiro, J., and Cortesao, L.: Discovering telecom fraud situations through mining anomalous behavior patterns. *In Proceedings of the DMBA Workshop on the 12th ACM SIGKDD*, 2006.

[Alves & Belo 07a] **Alves, R.**, Belo, O.: Effective Olap Mining of Evolving Data Marts. *IEEE IDEAS 2007*: to appear.

[Alves & Belo 07b] **Alves, R.**, Belo, O.: Mining Top-K Multidimensional Gradients. *DaWaK 2007*: to appear.

[Ayres et al. 02] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T.: 2002. Sequential PAttern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada, July 23 - 26, 2002). KDD '02. ACM Press, New York, NY, 429-435. DOI= http://doi.acm.org/10.1145/775047.775109

[Barbara et al. 97] Barbara, D., Sullivan, M.: Quasi-cubes: Exploiting Approximations in Multidimensional Databases. In Proc. Int. Conference on Management of Data (SIGMOD), 1997.

[Bayardo 98] Bayardo, R.: Efficiently Mining Long Patterns from Databases. In Proc. Int. Conference on Management of Data (SIGMOD), 1998.

[Berberidis et al. 04] Berberidis, C., Angelis, L., and Vlahavas, I.: 2004. PREVENT: An algorithm for mining intertransactional patterns for the prediction of rare events. In Proc. Second Starting AI Researchers' Symposium, volume 9 of Frontiers in Artificial Intelligence and Applications. IOS Press, 2004.

[Beyer & Ramakrishnan 99] Beyer, K., Ramakrishnan, R.: Bottom-Up Computation of Sparse and Iceberg CUBEs. SIGMOD 1999: 359-370.

[Bruno et al. 02] Bruno, N., Chaudhuri, S., Gravano, L.: Top-k selection queries over relational databases: Mapping strategies and performance evaluation. ACM Transactions on Database Systems, 2002.

[Burdick et al. 01] Burdick, D., Calimlim, M., Gehrke, J.: MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In Proc. Int. Conference on Data Engineering (ICDE), pp.443-452, 2001.

[Chakrabarti et al. 06] Chakrabarti, D., and Faloutsos. C.: Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.* 38, 1 (Jun. 2006), 2. DOI= http://doi.acm.org/http://doi.acm.org/10.1145/1132952.1132954.

[Chang et al. 00] Chang, Y., Bergman, L., Castelli, V., Li, M., L., C., Smith, J.: Onion technique: Indexing for linear optimization queries. In Proc. Int. Conference on Management of Data (SIGMOD), 2000.

[Chang et al. 05] Chang, C., Li, Y., and Lee, J.: 2005. An Efficient Algorithm for Incremental Mining of Association Rules. In Proceedings of the 15th international Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (Ride-Sdma'05) - Volume 00 (April 03 - 04, 2005).

[Cheung & Zaïane 03] Cheung, W., Zaïane, O. R.: Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint, Seventh International Database Engineering and Applications Symposium (IDEAS 2003), Hong Kong, China, July 16-18 (2003) 111-116.

[Cortes & Pregibon 01] Cortes, C., Pregibon, D.: Signature-based methods for data streams. *Data Mining and Knowledge Discovery*, (5):167-182, 2001.

[Cortes et al. 01] Cortes, C., Pregibon, D., and Volinsky, C.: Communities of Interest. In *Proceedings of the 4th international Conference on Advances in intelligent Data Analysis* (September 13 - 15, 2001). F. Hoffmann, D. J. Hand, N. M. Adams, D. H. Fisher, and G. Guimarães, Eds. Lecture Notes In Computer Science, vol. 2189. Springer-Verlag, London, 105-114.

[Cortes et al. 03] Cortes, C., Pregibon, D., and Volinsky, C.: Computational Methods for Dynamics Graphs, *Journal of Computational and Graphical Statistics* 12 (2003) (4), pp. 950–970.

[Dong et al. 04] Dong, G., Han, J., Lam, J., M., W., Pei, J., Wang, K., Zou, W.: Mining Constrained Gradients in Large Databases. IEEE Transactions on Knowledge Discovery and Data Engineering, 2004.

[Dong et al. 06] Dong, X., Han, J., Cheng, H., Xiaolei, L.: Answering Top-k Queries with Multi-Dimensional Selections: The Ranking Cube Approach. In Proc. Int. Conference on Very Large Databases (VLDB), 2006.

[El-Hajj & Zaïane 03] El-Hajj, M., Zaïane, O.R.: Inverted Matrix: Efficient Discovery of Frequent Items in Large Datasets in the Context of Interactive Mining, in Proc. 2003 Int'l Conf. on Knowledge Discovery and Data Mining (ACM SIGKDD), Washington, DC, USA, August 24-27 (2003) 109-118.

[Fang et al. 98] Fang, M., Shivakumar, N., Garcia-Molina, H., Motwani, R., Ullman, J., D.: Computing Iceberg Queries Efficiently. In Proc. Int. Conference on Very Large Databases (VLDB), 1998.

[Feng et al. 03] Feng, J., Si, H., and Feng, Y.: 2003. Indexing and incremental updating condensed data cube. In Proceedings of the 15th international Conference on Scientific and Statistical Database Management (Cambridge, MA, July 09 - 11, 2003).

[Feng et al. 04] Feng, Y., Agrawal, D., Abbadi, A. E., and Metwally, A.: Range CUBE: Efficient Cube Computation by Exploiting Data Correlation. In *Proceedings of the 20th international Conference on Data Engineering* (March 30 - April 02, 2004). ICDE. IEEE Computer Society, Washington, DC, 658.

[Ferreira et al. 05] Ferreira, P., **Alves, R.**, Azevedo, P., Belo, O.: A Hybrid Method to Discover Inter-Transactional Rules. In Proceedings of the *JISBD'2005*, Granada (2005)

[Ferreira et al. 06] Ferreira, P., **Alves, R.**, Belo, O. and Cortesao, L.: Establishing fraud detection patterns based on signatures. *In Proceedings of the 7th Industrial Conference on Data Mining*, 2006. DOI = 10.1007/11790853_41.

[Garofalakis et al. 99] Garofalakis, M. N., Rastogi, R., and Shim, K.: 1999. SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proceedings of the 25th international Conference on Very Large Data Bases* (September 07 - 10, 1999). M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, Eds. Very Large Data Bases. Morgan Kaufmann Publishers, San Francisco, CA, 223-234.

[Gouda & Zaki 05] Gouda, K., Zaki, J.: GenMax : An Efficient Algorithm for Mining Maximal Frequent Itemsets. Data Mining and Knowledge Discovery, 11:1-20, 2005.

[Gray et al. 96] Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. ICDE 1996: 152-159.

[Han & Kamber 06] Han, J. and Kamber, M.: 2006 Data Mining: Concepts and Techniques, 2nd ed. Morgan Kaufmann Publishers Inc.

[Han et al. 01] Han, J., Pei, J., Dong, G., Wank, K.: Efficient Computation of Iceberg Cubes with Complex Measures. In Proc. Int. Conference on Management of Data (SIGMOD), 2001.

[Han et al. 00] Han, J., Pei, J., Yin., Y.: Mining frequent patterns without candidate generation. In Proc. of ACM SIGMOD Intl. Conference on Management of Data, (2000) 1–12

[Han et al. 99] Han, J., Lakshmanan, L., Ng., R.: Constraint-Based Multidimensional Data Mining. IEEE Computer 32(8): 46-50 (1999).

[Hidber 99] Hidber, C.: Online association rule mining. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, volume 28(2) of SIGMOD Record. ACM Press (1999) 145–156

[Hristidis et al. 01] Hristidis, V., Koudas, N., Papakonstantinou, Y.: Prefer: A system for the efficient execution of multi-parametric ranked queries. In Proc. Int. Conference on Management of Data (SIGMOD), 2001.

[Imielinski et al. 02] Imielinski, T., Khachiyan, L., Abdulghani, A.: Cubegrades: Generalizing Association Rules. Data Mining and Knowledge Discovery, 2002.

[Kamber et al. 97] Kamber, M., Han, J., Chiang, J.: Metarule-Guided Mining of Multi-Dimensional Association Rules Using Data Cubes. KDD 1997: 207-210.

[Kim et al. 04] Kim, W.-Y., Lee, Y.-K., Han, J.: CCMine: Efficient Mining of Confidence-Closed Correlated Patterns. In Proc. Int. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2004.

[Kimball 02] Kimbal, R.: 2002. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd ed. Wiley.

[Kou et al. 04] Kou, Y., Lu, T., Sirwongwattana, S. and Huang, Y.: Survey of fraud detection techniques. In *Proceedings of IEEE Intl Conference on Networking, Sensing and Control*, March 2004.

[Lakshmanan et al. 02] Lakshmanan, J., Pei, J., and Han, J.: Quotient cube: How to summarize the semantics of a data cube. In VLDB'02, Hong Kong, China, Aug. 2002.

[Li et al. 04] Li, X., Han, J., Gonzalez, H.: High-dimensional OLAP: A Minimal Cubing Approach. In Proc. Int. Conference on Very Large Databases (VLDB), 2004.

[Lu et al. 00] Lu, H., Feng, L., and Han, J. 2000. Beyond intratransaction association analysis: mining multidimensional intertransaction association rules. *ACM Trans. Inf. Syst.* 18, 4 (Oct. 2000), 423-454. DOI= http://doi.acm.org/10.1145/358108.358114.

[Lunt 99] T.F. Lunt. A survey of intrusion detection techniques. *Computer and Security*, (53):405-418, 1999.

[Morfonios & Ioannidis 06] Morfonios, K. and Ioannidis, Y. 2006. CURE for cubes: cubing using a ROLAP engine. In *Proceedings of the 32nd international Conference on Very Large Data Bases* (Seoul, Korea, September 12 - 15, 2006). U. Dayal, K. Whang, D. Lomet, G. Alonso, G. Lohman, M. Kersten, S. K. Cha, and Y. Kim, Eds. Very Large Data Bases. VLDB Endowment, 379-390.

[Nanavati et al. 06] A. A. Nanavati, S. Gurumurthy, G. Das, D. Chakraborty, K. Dasgupta, S. Mukherjea, and A. Joshi. On the structural properties of massive telecom call graphs: findings and implications. In *Proceedings of the 15th ACM international Conference on information and Knowledge Management* (Arlington, Virginia, USA, November 06 - 11, 2006). CIKM '06. ACM Press, New York, NY, 435-444. DOI= http://doi.acm.org/10.1145/1183614.1183678.

[Navidi 2007] Navidi, C., W.: 2007. Statistics for Engineers and Scientists, 2nd ed. McGraw-Hill Science/Engineering/Math.

[Omiecinski 03] Omiecinski. Alternative Interest Measures for Mining Associations. IEEE Trans. Knowledge and Data Engineering, 15:57-69, 2003.

[Orlando et al. 01] Orlando, S., Palmerini, P., Perego, R.: Enhancing the apriori algorithm for frequent set counting. In Y. Kambayashi, W. Winiwarter, and M. Arikawa, editors, Proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery, volume 2114 of Lecture Notes in Computer Science (2001) 71–82.

[Orlando et al. 02] Orlando, S., Palmerini, P., Perego, R., Silvestri, F.: Adaptive and resource-aware mining of frequent sets. In V. Kumar, S. Tsumoto, P.S. Yu, and N.Zhong, editors, Proceedings of the 2002 IEEE International Conference on Data Mining. IEEE Computer Society (2002).

[Özden et al. 98] Özden, B., Ramaswamy, S., and Silberschatz, A. 1998. Cyclic Association Rules. In *Proceedings of the Fourteenth international Conference on Data Engineering* (February 23 - 27, 1998). ICDE. IEEE Computer Society, Washington, DC, 412-421.

[Pei et al. 00] Pei, J., Han, J. and Mao, P.: CLOSET: An efficient algorithm for mining frequent closed itemsets. In DMKD'00, May 2000.

[Pei et al. 01] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. 2001. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In *Proceedings of the 17th international Conference on Data Engineering* (April 02 - 06, 2001). IEEE Computer Society, Washington, DC, 215-224.

[Pennock et al. 99] Pennock, D. M., Flake, G. W., Lawrence, S., Glover, E. J. and Giles, C. L.: Winners Don't Take All: Characterizing the Competition for Links on the Web. *Proceedings of the National Academy of Sciences* 99, 8, 5207–5211.

[Rantzau 03] Rantzau, R.: Processing frequent itemset discovery queries by division and set containment join operators. In DMKD03: 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2003).

[Rymon 92] Rymon, R.: Search through Systematic Set Enumeration. In Proc. Int. Conference on Principles of Knowledge Representation and Reasoning (KR), 539-550, 1992.

[Sarawagi & Sathe 00] Sarawagi, S., Sathe, G.: i3: Intelligent, Interactive Investigaton of OLAP data cubes. In Proc. Int. Conference on Management of Data (SIGMOD), 2000.

[Sarawagi et al. 98a] Sarawagi, S., Thomas, S., Agrawal, R.: Integrating mining with relational database systems: alternatives and implications. In Proc. of the ACM SIGMOD Conference on Management of data, Seattle, Washington, USA (1998).

[Sarawagi et al. 98b] Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-Driven Exploration of OLAP Data Cubes. EDBT 1998: 168-182.

[Sathe & Sarawagi 01] Sathe, G., Sarawagi, S.: Intelligent Rollups in Multidimensional OLAP Data. In Proc. Int. Conference on Very Large Databases (VLDB), 2001.

[Shang et al. 04] Shang, X., Sattler, K., Geist, I.: Sql based frequent pattern mining without candidate generation. In SAC'04 Data Mining, Nicosia, Cyprus (2004).

[Shao et al. 04] Shao, Z., Han, J., Xin, D.: MM-Cubing: Computing Iceberg Cubes by Factorizing the Lattice Space. In Proc. Int. Conference on Scientific and Statistical Database Management (SSDBM), 2004.

[Sismanis et al. 02] Sismanis, Y., Deligiannakis, A., Roussopoulos, N., and Kotidis, Y. 2002. Dwarf: shrinking the PetaCube. In *Proceedings of the 2002 ACM SIGMOD international Conference on Management of Data* (Madison, Wisconsin, June 03 - 06, 2002). SIGMOD '02. ACM Press, New York, NY, 464-475. DOI= http://doi.acm.org/10.1145/564691.564745

[Spiliopoulou 99] Spiliopoulou, M. Managing interesting rules in sequence mining. In Proc. European Conf. on Principles and Practice of Knowledge Discovery in D atabases, 554-560, 1999.

[Srikant & Agrawal 96] Srikant, R. and Agrawal, R. 1996. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th international Conference on Extending Database Technology: Advances in Database Technology* (March 25 - 29, 1996). P. M. Apers, M. Bouzeghoub, and G. Gardarin, Eds. Lecture Notes In Computer Science, vol. 1057. Springer-Verlag, London, 3-17.

[Tung et al. 03] Tung, A. K., Lu, H., Han, J., and Feng, L. 2003. Efficient Mining of Intertransaction Association Rules. *IEEE Transactions on Knowledge and Data Engineering* 15, 1 (Jan. 2003), 43-56. DOI= http://dx.doi.org/10.1109/TKDE.2003.1161581

[Wang & Zaniolo 00] Wang, H., Zaniolo, C.: Using SQL to build new aggregates and extenders for Object-Relational systems. In Proc. Of the 26th Int. Conf. on Very Large Databases, Cairo, Egypt (2000).

[Xin et al. 06] Xin, D., Shao, Z., Han, J., Liu, H.: C-Cubing: Efficient Computation of Closed Cubes by Aggregation-Based Checking. ICDE 2006: 4.

[Xiong et al. 03] Xiong, H., Tan, P.-N., Kumar, V. Mining Strong Affinity Associations Patterns in Data Sets with Skewed Support Distribution. In Proc. Int. Conference on Data Mining (ICDM), 2003.

[Yoshizawa 00] Yoshizawa, T., Pramudiono, I., Kitsuregawa, M.: Sql based association rule mining using commercial rdbms (ibm db2 udb eee). In In Proc. DaWaK, London, UK (2000).

[Zaki & Hsiao 02] Zaki, M. J. and Hsiao, C.: CHARM: An efficient algorithm for closed itemset mining. In SDM'02, April 2002.

[Zaki 01] Zaki, M. J. 2001. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Mach. Learn.* 42, 1-2 (Jan. 2001), 31-60.

[Zhao et al. 97] Zhao, Y, Deshpande, P, Naughton, J: An Array-Based Algorithm for Simultaneous Multidimensional Aggregates. SIGMOD 1997: 159-170.

[Zou et al. 02] Zou, Q., Chu, W.-W., Lu, B.: SmartMiner: A Depth First Algorithm Guided by Tail Information for Mining Maximal Frequent Itemsets. In Proc. Int. Conference on Data Mining (ICDM), 2002.