Actes des 17ème Journées Bases de Données Avancées BDA'01, Agadir (Maroc),
Octobre 2001. Cepadues Editions. pp. 311-323.

Query driven knowledge discovery via OLAP manipulations

Jean-François BOULICAUT[1], Patrick MARCEL[2] et Christophe RIGOTTI[1]

[1] INSA Lyon, LISI bât. 501, F-69621 Villeurbanne, France
[2] Université F. Rabelais, LI, 3 place J. Jaurès, F-41000 Blois, France

E-mail : {jfboulic, crigotti}@lisi.insa-lyon.fr, marcel@univ-tours.fr

Résumé

Nous étudions sous l'angle langage de requêtes *la spécification des processus d'extraction de connaissances dans les données contenues dans les systèmes OLAP, c'est à dire des données par nature multidimensionnelles et pouvant se consulter à plusieurs niveaux de granularité. Dans ce travail nous considérons plus particulièrement les processus centrés sur la découverte de règles d'associations. Nous utilisons un langage requêtes permettant de spécifier les manipulations de données dans les systèmes OLAP et nous montrons qu'il est aussi adapté à la description des traitements de pré-processing et post-processing nécessaires lors de la recherche de règles d'associations. Ce langage de requêtes et le modèle de données sous-jacent fournissent un support homogène de description du processus d'extraction de connaissances et permettent ainsi de mieux appréhender le caractère interactif d'un tel processus.*

Mots clés : *règles d'associations, OLAP, langage de requêtes*

Abstract

We study KDD (Knowledge Discovery in Databases) processes on OLAP (multidimensional and multilevel) data from a query point of view. Focusing on association rule mining, we consider typical queries to cope with the pre-processing of multidimensional data and the post-processing of the discovered patterns as well. We use a model and a rule-based language stemming from the OLAP representation and manipulation, and argue that such a language fits well for writing KDD queries on multidimensional and multilevel data. Using an homogeneous data model and our language for expressing queries at every phase of the process appears as a valuable step towards a better understanding of interactivity during the whole process.

Keywords : *association rules, OLAP, query language*

1 Introduction

Discovering knowledge from data appears as a complex iterative and interactive process containing many steps : from the preparation of the data set (also called pre-processing), to the post-processing of discovered patterns [14]. Different kinds of patterns might be used and therefore different data mining techniques are needed (e.g., association and episode rules for alarm analysis, clusters and decision trees for sales data analysis). This paper addresses the challenge of supporting KDD processes using a querying approach.

Following Imielinski and Mannila [22], second generation data mining systems might support the discovery process by means of powerful query languages. Indeed, managing the whole KDD process is one of the crucial issues to design methodologies and integrated environments for relevant classes of applications. This is a long term goal for which one can not expect a comprehensive answer in the near future.

As a valuable step towards such a goal, we study how a query language dedicated to data manipulation can be used during the process of association rule mining [2]. We choose not to concentrate on the data mining phase since impressive results have been obtained in the last years for this phase (e.g., the efficient discovery of frequent sets in binary data), while less attention has been given to the pre-processing and post-processing phases.

These interactive non trivial tasks are often ad-hoc and simply done by means of standard database queries and/or programming scripts. We argue that these steps should be performed with a single model and query language so that raw data and mined patterns can be accessed uniformly, thus giving rise to static analysis and optimization of these steps. Therefore, we propose a model and a language for expressing queries at different steps of a KDD process. This language stems from On-Line Analytical (OLAP) representation and manipulation where a *cube* model is used to organize and manipulate data according to multiple perspectives and multiple granularities.

Several reasons motivate this proposal :

– data mining is often seen as a client of OLAP servers [4, 12],
– there is a natural correspondance between OLAP treatments and some operations performed during pre-processing and post-processing [8],
– OLAP treatments are also concerned with interactive data analysis [20], and
– there is no loss of generality since standard transactional processing can be expressed with an OLAP language [4, 26].

The context is as follows :

– the data to be mined (called raw data) are contained in multidimensional cubes or can be put under the form of cubes. Moreover, each dimension of a cube can be associated with a hierarchy,
– we are interested in association rule mining,
– the frequent sets from which association rules are derived, are discovered by using the Apriori algorithm [3] and its extensions or better algorithms that have been designed recently like [28, 6, 21, 29, 9, 11].

Under these assumptions, the pre-processing step is a collec-

tion of queries on raw data that provide binary tables. These binary tables are input to the mining algorithms that provide the frequents sets. The post-processing step turns to be a collection of queries on raw data and on the results of the mining step to provide interesting patterns (i.e., rules) from the extracted frequent sets.

Main contributions This paper is a significant extension of [8] by pointing out the complementarity between OLAP treatments and the support of KDD processes. We show that an OLAP data model enables to handle the description of multilevel and multidimensional raw data (called OLAP data in the following), binary tables, collections of frequent itemsets and frequent association rules. We show that OLAP queries are well suited for expressing some important non trivial treatments within typical rule mining processes.
Providing a framework for reasoning about sequence of queries to define complex mining processes is a long-term goal. This ongoing research investigates rather simple but yet not well understood issues about association rule mining. Even if we do not address implementation issues, we emphasize that OLAP systems can be straightforwardly extended to incorporate some association rule extraction facilities, providing new analysis tool for data owners.

Outline This paper is organized as follows. In Section 2 we present our motivations for connecting rule mining processes and OLAP possibilities. Generalities about the KDD process, the rule mining framework and OLAP concepts are given. Section 3 introduces informally the data model and the manipulation language we use. Section 4 illustrates via examples the use of an OLAP query language during association rule mining. We conclude in Section 5.

2 Preliminaries : Queries for rule mining

The goal of this section is to survey the needs in terms of queries for typical rule mining processes. We begin by a a short description of some KDD concepts, focusing on the rule mining framework. Then we briefly present OLAP data manipulation. We conclude by motivating the connections between rule mining and OLAP queries.

Knowledge Discovery in Databases Discovering knowledge from data is a complex process containing many steps (e.g., [14]) :

1. Understanding the data, that is the domain background knowledge that guides the whole discovery process.
2. Pre-processing the data, i.e., preparing the data set for a mining task. The first thing to do is to collect the data to be mined (the raw data). These data may be under different forms (e.g., databases, flat files), and so must be put into a form suitable to the mining algorithm. This is done by e.g., selecting relevant attributes, cleaning, sampling, grouping and/or discretizing data. We find here, among others, the major technical issues of data warehouse design.
3. Discovering potentially interesting patterns (the mining phase) i.e., using an extraction algorithm on the prepared data set.
4. Post-processing of discovered patterns, which consists in dealing with the (often huge) amount of discovered patterns to highlight the most significant ones once the end-user needs are defined explicitely. This step relies on objective and subjective measures of interestingness. Objective measures can be computed for each pattern and enable ordering (e.g., ranking the rules w.r.t. confidence measure). Subjective measures are often user-driven queries based on more or less sophisticated templates [24, 5].
5. Putting the results in use, that is obtaining a user friendly representation where simple and intuitive manipulations (e.g., selection, visualization) are allowed. It concerns also more or less automatic ways to use discovered rules to produce operational business rules.

Moreover, a KDD process is highly *iterative and interactive*. Discovering patterns leads to data pre-processing refinement and the defintion of a new mining tasks that in turns provide patterns that have to be post-processed, etc.
As a prototypical example of a KDD process, we consider the framework of association rule mining.

Rule mining framework The association rule mining problem has received much attention since its introduction in [2]. Given a schema $R = \{a_1, \ldots, a_n\}$ of attributes with domain $\{0, 1\}$, and a relation r over R, an $association\ rule$ about r is an expression of the form $X \Rightarrow Y$, where $X \subseteq R$ and $Y \in R \setminus X$. The intuitive meaning of the rule is that if a row of the matrix r has a 1 in each column of X, then the row tends to have a 1 also in column Y. This semantics is captured by $support$ and $confidence$ values. Given $W \subseteq R$, $support(W, r)$ denotes the fraction of rows of r that have a 1 in each column of W. W is called a frequent set if its support is greater than some user-given support threshold. The frequency of $X \Rightarrow Y$ in r is defined to be $support(X \cup \{Y\}, r)$ while its confidence is $support(X \cup \{Y\}, r)/support(X, r)$. Typically, we are interested in association rules for which the frequency and the confidence are greater than user-given thresholds.
Frequent sets are a concise representation for binary data from which association rules are easily derived [3]. The intuition is that for every frequent set X, one can test for the interestingness (e.g., confidence is greater than the user-given threshold) of the rule $X \setminus Y \Rightarrow Y$ where $Y \in X$. Though an exponential search space is concerned, association rules can be computed thanks to the thresholds on one hand and a safe pruning criterion that drastically reduces the search space on the other hand (the so-called *apriori* trick [3]).
In this case of association rule mining, the pre-processing of raw data implies the building of the binary relevant representation r (relevancy w.r.t. the kind of properties the user is interested in). The post-processing of the extracted patterns implies to derive "interesting" association rules (interestingness w.r.t. objective measures like confidence [2], conviction [10], intensity of implication [17], J-measure [31] or subjective measures like templates [24]).

2.1 On-Line Analytical Processing

This section introduces the concept of On-Line Analytical Processing (OLAP). The reader interested in a complete overview can refer to [12, 26].

The need for OLAP stems from the evolution of commercial databases from transactional to decision support applications. E.F. Codd [4] identifies the category of OLAP treatments as the ones used in different front-ends for data analysis purpose (e.g., spreadsheets, graphical interfaces). These treatments are concerned with the description and the manipulation of data according to *multiple perspectives* and *multiple granularities*.

Multiple perspectives To fit the needs of user-analysts, business historical data should reflect the natural view of the enterprise. The intuition is to consider data as points in a multidimensional space. Indeed the user may want to represent the information under the convenient form of a *cross-tab*, where each point in the 2-dimensional space $customers \times items$ is associated with a value $amount$. For example, Figure 1 displays the number of items purchased by different customers.

sales	wine	beer	bread	meat
kate	0	20	50	20
mike	70	0	0	60
rick	0	50	50	20

FIG. 1 – The bidimensional table *sales*

The *cube* metaphor has emerged to generalize this conceptual view of multidimensional data. Figure 2 depicts a cube based on a typical example of the OLAP literature, that can be seen as the extension of the table of Figure 1 to the $time$ dimension.

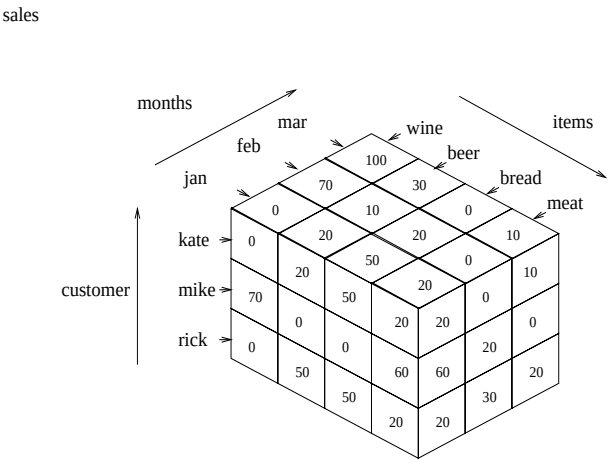

FIG. 2 – The cube *sales*

A cube is composed of *cells* containing one or several values called *measures*. Each axis of the cube corresponds to a dimension, and is graduated by values called *members*. The location of a cell is given by one member of each axis, and is called a *reference*. Hence a reference identifies in a unique manner a cell of a cube.

In addition to standard relational operations (e.g., selection, projection) that can be easily extended to cube manipulation, several elementary[1] operations are concerned with the structural aspect of a cube. They are refered to as *restructuring* operations, as they are used to animate this cubic representation and provide multiple perspectives. It is to be noticed that each restructuring operation admits a reciprocal operation. We give the intuitions underlying the operations and we give, for each operation, an example of the result of this operation :

- *rotate* means to reorder the axes of the multidimensional view (Figure 3 depicts a rotate applied to the table of Figure 1) ;
- *split* means to present separately each "slice" of the cube (Figure 4 depicts a split applied to the cube of Figure 2) ;
- *nest/unnest* allows to nest/unnest members of different dimensions (Figure 5 depicts a nest applied to the cube of Figure 2) ;
- *push/pull* pushing consists in combining members of a cube to measures. Pulling, the reciprocal operation, allows to set up a new dimension from measures. These two operations provide *symmetric treatment* of members and measures (Figure 6 depicts a pull applied to the table of Figure 1, where the constant 1 is used to represent the validity of the combination of each member, and the constant 0 is used to represent the invalidity of the combination of each member).

sales	kate	mike	rick
wine	0	70	0
beer	20	0	50
bread	50	0	50
meat	20	60	20

FIG. 3 – "Rotate"

Multiple Granularities An other aspect of enterprise data is that it can be viewed at different levels of detail, called *granularity levels* in the OLAP literature. Indeed, each dimension of a cube is associated with a graph that indicates how members of a level are *grouped* to form the members of the immediate ancestor level. An example of such graphs, called *hierarchies* is given in Figure 7. These graphs describe the different granularity levels of the cube $sales$ of Figure 2, and indicate, for example, how items are grouped into subcategories. Usually, the name of the top level is the name of the dimension.

The elementary operations dealing with granularity specify the relative navigation along grouping hierarchies (we give for each operation an example of the result of the operation applied to the cube $sales$ of Figure 2) :

- *rolling-up* means to increase the level of aggregation, i.e., to decrease the level of detail (Figure 8 depicts such a roll-up to sum up the sales over the years) ;
- *drilling-down* means to decrease the level of aggregation, i.e., to increase the level of detail (Figure 9 depicts a drill down to present the sales results at the cities level).

[1] Note that the term elementary refers to the user's need [12, 30, 4].

sales wine	jan	feb	mar
kate	0	70	100
rick	0	10	10
mike	70	50	40

sales beer	jan	feb	mar
kate	20	10	30
rick	50	50	50
mike	0	10	40

sales bread	jan	feb	mar
kate	50	20	0
rick	50	60	60
mike	0	10	0

sales meat	jan	feb	mar
kate	20	0	10
rick	60	30	20
mike	40	20	0

FIG. 4 – “Split”

nested sales		jan	feb	mar
kate	wine	0	70	100
	beer	20	10	30
	meat	20	0	10
	bread	50	20	0
rick	wine	0	10	10
	beer	50	50	50
	meat	60	30	20
	bread	50	60	60
mike	wine	70	50	40
	beer	0	10	40
	meat	40	20	0
	bread	0	10	0

FIG. 5 – “Nest”

2.2 Connections between rule mining and OLAP querying

This section motivates the use of an OLAP query language for expressing parts of the rule mining processes. Let us first motivate the need for a query language during the KDD process. As pointed out above, pre-processing and post-processing are non trivial tasks, even in the simple case of association rule mining.

– It is interactive : today most of the pre-processing and post-processing tasks are often done by means of ad-hoc programming scripts (like e.g., `perl` or `awk`) on flat files. A query language is the opportunity to cleanly interact with a database and to express these tasks under a readable and reusable form. Properties of the query language can also be exploited for static analysis.
– It is iterative : refining the KDD process implies comparing the results of the post-processing phase with what has been done during the pre-processing phase. It leads to possibly new definitions of input data for further mining tasks (the binary table in the case of association rules). With a single query language and data model used in both phase, raw data and mined patterns can be accessed uniformly, and the different tasks of these phases can be optimized.

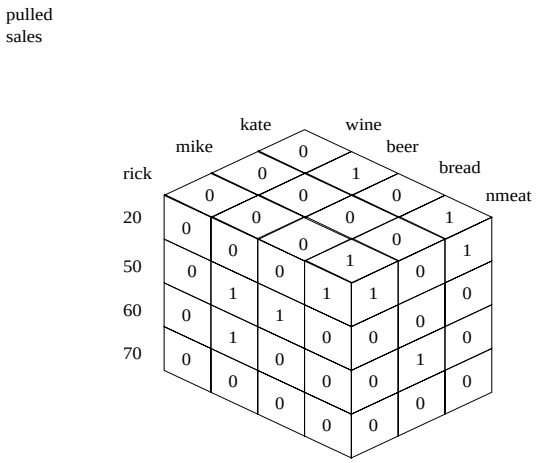

FIG. 6 – “Pull”

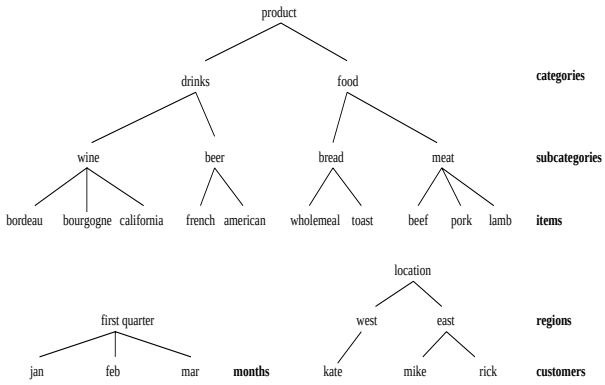

FIG. 7 – Grouping hierarchies

[16] adresses this issue by presenting an approach with OQL as a query language. Dedicated languages like [27, 23] also enable to select data and specify various mining tasks. We already have shown [8] that a multidimensional query language is a good candidate to express tasks of the pre-processing and post-processing phase. Actually, the idea of integrating rule mining and OLAP stems from Han [20]. Indeed, OLAP is query driven, and the model provided by OLAP allows interactive and flexible data analysis. Since KDD is machine driven, OLAP and KDD can be seen as complementary. The data cube structure facilitates efficient mining of multilevel and multidimensional association rules.

Despite these advances, there is still a lack of a clear understanding of the pre-processing and the post-processing of multilevel and multidimensional OLAP data. Existing approaches do not consider the correspondance of OLAP operations and KDD process tasks, and they do not consider the multidimensional and multilevel aspect of data and patterns. As a consequence no formal framework has been proposed for integrating rule mining and OLAP at a logical level.

We propose to work in this direction. One of the goals of this paper is to emphasize several strong connections between

firts quarter sales	wine	beer	meat	bread
kate	170	60	30	70
mike	160	50	60	10
rick	20	150	110	170

FIG. 8 – "Roll-up"

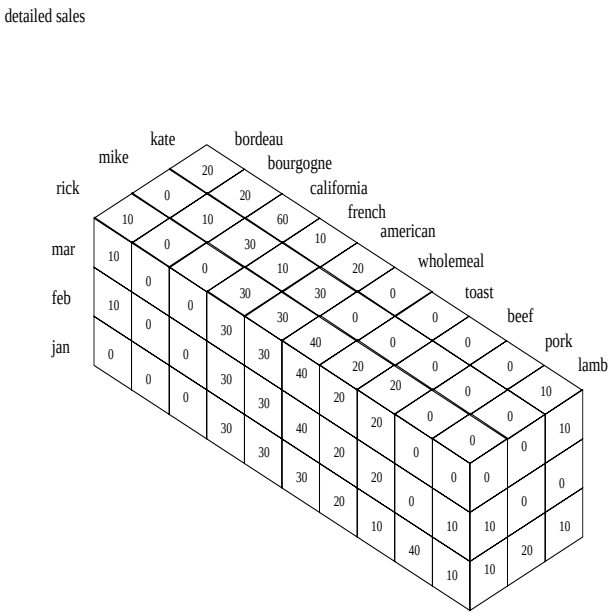

FIG. 9 – The cube *detailed sales*

OLAP manipulations and parts of association rule mining processes. This clearly extends [8] by taking aggregation on hierarchical data into account.

3 Data model and language overview

The goal of this section is to motivate and describe the use of a particular OLAP language for expressing queries during the KDD process. The query language we use is not the main contribution of this work and we do not provide here its formal semantics nor implementation techniques. It is clearly out of the scope of this paper. However, when details on the language are available, we refer to the relevant papers.

Among the existing formal query languages for dealing with OLAP data manipulation at a logical level, (see e.g., [26]) we choose to use an extension of Datalog for the sake of readability. Indeed, this language can be used to express complex OLAP queries in a concise manner. As an extension of Datalog, it can also be used to express relational queries, as well as queries for putting relations under the form of cubes. It is introduced in [19, 18], and the reader could refer to these references for more details. It should be noticed that every program in this language is translatable into standard Datalog.

The language and its data model are now informally presented. Throughout the presentation, the reader is assumed to be familiar with Datalog [1].

3.1 Data model

Names The constants we use in the model are called *atomic names*. Structured names can be built from atomic names using the constructor ".", and are called *nested names*. In the following, we use the term *names* to denote both atomic and nested names.

Cells In this multidimensional data model, data are organized in *cells*. A cell is identified by a *cell reference*, and is associated with a unique *cell contents*. A cell reference is of the form $N(N_1, N_2, \ldots, N_p)$, where $N, N_1, N_2, \ldots, N_p$ are names. N is the $cube$ name, and $N_1, N_2, \ldots, N_p$ are the *members* (i.e., coordinates in a p-dimensional space).
A cell contents is a tuple of names. Associations of cells contents with cells references are represented by *ground atoms* of the form :

$$N(N_1, N_2, \ldots, N_p) : \langle N_{p+1}, \ldots, N_{p+q} \rangle$$

where $N_{p+1}, \ldots, N_{p+q}$ are measures. We call this form of atoms *cell atoms*.

Cubes A *cube* is a set of ground cell atoms having both a common cube name and the same number of members, and in which the same reference does not appear more than once to ensure cell monovaluation.

Database A *multidimensional database* is a set of ground cell atoms in which the same reference does not appear more than once.

example Consider the cube $sales$ of Figure 2. This cube is represented by the following set :

$$\{sales(kate, jan, wine) : \langle 0 \rangle,$$
$$\vdots$$
$$sales(mike, feb, meat) : \langle 20 \rangle\}$$

Consider now the cube of Figure 5, which represents a nested organization of the data of cube *sales*. Using nested names, this cube, called *nested sales* is described by the set :

$$\{nestedSales(kate.wine, jan) : \langle 0 \rangle,$$
$$\vdots$$
$$nestedSales(mike.meat, feb) : \langle 20 \rangle\}$$

3.2 Intuitive syntax and semantics of the language

In this subsection, we present informally the syntax and the semantics of a rule-based language, and show how it can express the elementary OLAP operations.
Rules à la Datalog are used in this language to define new cell references and their associated contents from existing cells. A higher order syntax stemming from Hilog [13] allows variables to range over atomic names used in cell references and in cell contents. It should be noted that variables cannot range over nested names but only over atomic names.
We adopt the following conventions : symbols beginning with an upper-case letter denote variables, and symbols beginning with a lower-case letter or a digit denote constants. Built-in predicates (e.g., $\leq$) and standard Datalog predicates are allowed. "_" denotes an anonymous variable.

Intuitive meaning Consider the rule $p(X) \longleftarrow q(X, Y), r(Y)$. The standard (Datalog) informal meaning of this rule

is *if q(X,Y) holds and r(Y) holds, then p(X) holds.* The basic intuition of our extension is to read such a rule as follows : *if there are two cells of references q(X,Y) and r(Y), then there is a cell of reference p(X).* We also add the handling of cell contents, and then a typical rule will be :

$$p(X) : \langle W \rangle \longleftarrow q(X,Y) : \langle W \rangle, r(Y) : \langle X \rangle.$$

This rule will be informally read : *if there exists a cell of reference q(X,Y) containing W, and if there exists a cell of reference r(Y) containing X, then there exists a cell of reference p(X) containing W.*

3.3 OLAP elementary data manipulations

We now illustrate how rules can specify the elementary data manipulations used in OLAP systems on the cubes of Figure 1 and 2.

Restructuring cubes We first show how the cube restructuring operations can be trivially specified.

Rotate The following rule can be used to rotate the table $sales$ and obtain the table depicted in figure 3 :

$$rotatedSales(I,C) : \langle S \rangle \longleftarrow sales(C,I) : \langle S \rangle.$$

Split Suppose we want to split the cube $sales$ in order to obtain a cube per item (Figure 4). We can use a variable ranging over items to form new cube names. This is shown by the following rule :

$$sales.I(C,M) : \langle S \rangle \longleftarrow sales(I,C,M) : \langle S \rangle.$$

Nest The nested representation of Figure 5 can be obtained from the cube $sales$ by the rule :

$$nestedSales(C.I,M) : \langle S \rangle \longleftarrow sales(I,C,M) : \langle S \rangle.$$

Push, pull Suppose we want to pull the measures from the cube $januarySales$ (Figure 1) to obtain the cube depicted in Figure 6. This pull operation can be expressed by the following rules :

$$pulledSales(C,S,I) : \langle 1 \rangle \longleftarrow sales(C,I) : \langle S \rangle.$$
$$pulledSales(C,S,I) : \langle 0 \rangle \longleftarrow not(sales(C,I) : \langle S \rangle), sales(C,\_) : \langle \_ \rangle, sales(\_,I) : \langle \_ \rangle, sales(\_,\_) : \langle S \rangle.$$

Summaries specification We now present how rules can be used to specify summarizations of data at different levels of aggregation. The following examples are based on the 2-dimensional cube named $detailedSales$ depicted in Figure 9.

The specification of summaries requires that the grouping relationship between members is known (e.g., how customers are grouped into regions). Such a relationship is depicted in Figure 7. In this example, customers can be grouped into regions, and regions can be grouped to form the whole area called *location*. In the same way, the different items can be grouped to form the whole production called *product*.

The grouping relationship is represented by particular literals of the form $in(\alpha, \beta)$, where α and β are atomic names. For example the grouping relationship of Figure 7 is described by the following set of ground grouping atoms : $\{in(beer, drinks), \ldots, in(kate, west), \ldots, in(west, location)\}$

It is to be noted that the grouping relationship can be specified by rules.

Aggregate subgoals Aggregate subgoals used to specify the summaries are of the form : $T = f(N(N_1, \ldots, N_p))$ where T is a constant or a variable, f is an aggregate operator (e.g., *sum, min*) and $N(N_1, \ldots, N_p)$ is a possibly non ground cell reference[2].

Their intuitive meaning is illustrated by the following example. Consider the ground aggregate subgoal :

$$60 = sum(detailedSales(rick, mar, bread)).$$

It holds if the amount of *bread* bought by *rick* in *march* is equal to 60. More precisely, let $detailRef(detailedsales(rick,mar,bread))$ be the set of the references corresponding to the lowest level of description for rick's puchases of bread in march. According to the grouping relationship depicted in Figure 7, we have :

$$detailRef(detailedSales(rick, mar, bread)) = \{detailedSales(rick, mar, wholemeal), detailedSales(rick, mar, toast)\}.$$

Assume $detailCont(detailedSales(rick, mar, bread))$ is the multiset formed with contents of the references in the set $detailRef(detailedSales(rick, mar, bread))$. Then, we have : $detailCont(detailedSales(rick, mar, bread)) = \{\langle 40 \rangle, \langle 20 \rangle\}$.

The semantics of the aggregate subgoal can now be stated more precisely : $60 = sum(detailedSales(rick, mar, bread))$ holds if the sum of the elements in the multiset $detailCont(detailedSales(rick, mar, bread))$ is equal to 60.

Let us now illustrate the use of aggregate subgoals for expressing roll-ups, on the cubes *sales* and *detailedSales* using the grouping relationship depicted in Figure 7.

Simple roll-ups Suppose we want to roll-up the data of the cube $sales$ to obtain a cube containing the amount of *wine* purchased by $kate$ during the first quarter. We simply use the following rule :

$$total(kate, first\ quarter, wine) : \langle T \rangle \longleftarrow T = sum(sales(kate, first quarter, wine)).$$

It is to be noted that, following the same principle, the cube $sales$ can be obtained by rolling-up the data of cube $detailed\ sales$ (Figure 9) to the different categories (we use the in relation to restrict the variables C and I to range respectively over customers and items) :

$$sales(C,M,I) : \langle T \rangle \longleftarrow T = sum(detailedSalesCities(C,M,I)), in(C,X), in(X, location), in(I,Y), in(Y, product), in(M, first\ quarter).$$

Multiple roll-up Computing the total sales by categories from the cube *sales* can be done by rolling-up simultaneously from the months level to the first quarter level, and from the customers level to the location level :

$$totalsales(I, total) : \langle T \rangle \longleftarrow T = sum(sales(location, first quarter, I)), in(I, product).$$

A graphical counterpart of the cube $total\ sales$ is represented in Figure 10.

[2]Note that rules are range restricted [1], i.e., every variable appearing in an aggregate subgoal must also appear in an other literal of the body of the

total sales	total
drinks	610
food	450

FIG. 10 – The cube *total sales*

Relational operations Classical relational operations are of course still needed in OLAP query languages. Suppose we are interested in a subset of the cube $sales$, that contains only the data for january (to obtain the table of Figure 1). Such a table can be obtained with a combination of selection and projection called *slicing-dicing* in the OLAP literature. This combination can be expressed using the following rule, that selects the month january, and projects on the customer and items dimensions :

$sales(C, I) : \langle S \rangle \longleftarrow sales(C, mar, I) : \langle S \rangle.$

The other relational operations are taken into account straightforwardly as in Datalog [1].

4 OLAP queries in the KDD process

This section presents typical treatments during an association rule mining process. It concerns both the pre-processing and the post-processing phases. These treatments can be considered as OLAP queries, and thus can be expressed with the language introduced in the previous section. We begin by describing how we model the association rule process, and next we present some typical pre-processing and post-processing queries.

4.1 Modeling the rule mining process

The way we propose to model the extraction of association rules can be summarized by Figure 11. The result of the pre-processing step is a collection of binary tables that are the input of the mining algorithm. The binary tables have attributes as row names and attributes as column names. Examples of binary tables are depicted in Figures 13 and 14. The output of the mining step is a collection of tables representing the frequent item sets mined on the corresponding $input$ tables. The $frequents$ tables have attributes as column names and frequent set identifiers as row names. They are binary tables as well : a cell contains 1 if the frequent set (whose identifier is the row name) contains the attribute as column names. A column that contains the support for each frequent set is added. Examples of $frequents$ tables are depicted in Figures 15 and 16.

The data model leads us to take the following conventions :

– a naming convention for binary tables : nested names are used for differenciating them.
– when constructing a binary table, every row or column attribute has the same level of nesting. Otherwise, post-processing queries like inclusion would be less easy to specify.

Association rules are derived from frequent sets and represented in tables too. The form we choose is the following (but OLAP queries can be used to restructure the $rules$ tables to derive more user-friendly representations) : the row name refers to the body of the rule and the column name refers to the head. If this rule holds in the raw data, then the cell contains its support and its confidence. It might also contain other interestingness measures (e.g., conviction) associated with this rule. This model emphasizes the role of frequent sets as a condensed representation from which various knowledge elicitation can be derived [25].

rule.

4.2 Pre-processing queries

The main task the user has to perform during the pre-processing step is to filter and transform the raw data so that (a) all the relevant information from which knowledge is to be explicited is available and, (b), it satisfies the input requirements of the mining algorithms. In this paper, we consider only pre-processing steps that provide binary tables. We assume that, provided such a binary table, data owners can assign a meaning to frequent itemsets in it and thus interpret the derived association rules. Hence a pre-processing step consists in using OLAP queries to construct binary tables. Generalizing that, we can describe the pre-processing phase as a collection of queries that transform the raw data into a set of binary tables (each being an input to the mining algorithm to perform various mining tasks).

We describe how to get binary tables from the table $cust$ of Figure 12 and the cube $sales$ of Figure 2. It illustrates how queries express typical pre-processing steps.

cust	age	income	lives in
kate	20	3000	west
mike	40	2000	east
rick	80	1000	east

FIG. 12 – The table $cust$

Binary tables In our model, a binary table can be defined as a table whose cells contain either the constant 1 or the constant 0. 2 binary tables are depicted in Figure 13 and Figure 14. Recall that cells from the cube $sales$ are of the form $sales(C, M, I) : S$ where C is a customer, M is a month, I is an item and S is the number of item I bought by C in M. Some of the typical OLAP operations (e.g., nest, pull) are very useful to change the structure of cubes and tables, and can be used to construct the binary table(s) from raw data. In the following, we suppose that the user does not want to represent the number of items sold, then a nest operation and a selection can be used to obtain a binary table $input.cust$ from the cube $sales$:

$input.cust(C, M.I) : 1 \longleftarrow sales(C, M, I) : S, S > 0.$
$input.cust(C, M.I) : 0 \longleftarrow sales(C, M, I) : 0.$

Expressing background knowledge The queries can be used to incorporate the background knowledge (knowledge not to be discovered but not explicitly represented in the data).

Suppose the user wants to describe the fact that a customer is a good customer if he/she is a regular customer and has an income over 2000. This can be done by the following

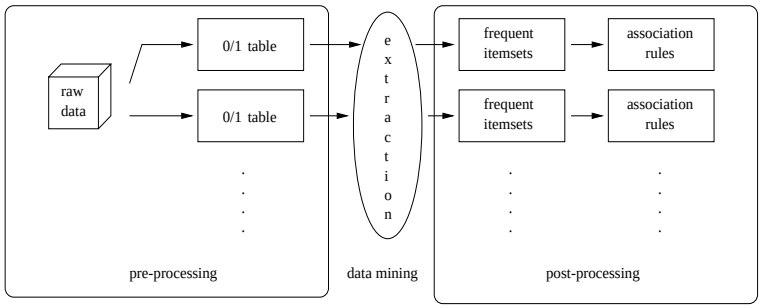

FIG. 11 – Modeling the KDD process

query, where $regularCustomer$ is a table containing previously known (or extracted) information :

$$cust(C, type) : good \longleftarrow cust(C, income) : I, I > 2000, regularCustomer(C).$$

Aggregation (roll-up) can also be used to express background knowledge. For example, a very good customer can be defined as a regular customer who purchased more than 150 items in the first quarter :

$$cust(C, type) : verygood \longleftarrow X = sum(sales(C, firstquarter, I)), X > 150, regularCustomer(C).$$

Filtering Since the OLAP language can express classical relational operations (e.g., selection, projection, join, difference), OLAP queries can be used to discard unwanted attributes. Suppose the user is only interested in very good customers. The query for constructing the binary table becomes :

$$input.cust(C, M.I) : 1 \longleftarrow sales(C, M, I) : S, S > 0, \text{cust(C,type) : very good.}$$
$$input.cust(C, M.I) : 0 \longleftarrow sales(C, M, I) : 0, cust(C, type) : verygood.$$

Discretizing Raw data often contain continuous valued attributes (e.g., age, salary). It is common to discretize these attributes (i.e., to group them into intervals) to ensure that they reach a sufficient level of support. For example, assume the user wants the binary table to contain only two categories of age :

$$input.cust(C, age.lt\_50) : 1 \longleftarrow cust(C, age) : A, A < 50.$$
$$input.cust(C, age.lt\_50) : 0 \longleftarrow cust(C, age) : A, A \geq 50.$$
$$input.cust(C, age.ge\_50) : 1 \longleftarrow cust(C, age) : A, A \geq 50.$$
$$input.cust(C, age.ge\_50) : 0 \longleftarrow cust(C, age) : A, A < 50.$$

Indeed, the decision to split on the value 50 is the hardest part of that step. This is a well-identified problem in data analysis and discussing it is out of the scope of this paper.

Selecting A rotate operation can be used on the binary table to select which kind of association rules are to be extracted. For example, in the table $input$ of Figure 13, association rules will concern each customer behavior. If the user

	age		jan		...
input cust	lt_50	ge_50	wine	beer	...
kate	1	0	1	1	...
mike	1	0	1	0	...
⋮	⋮	⋮	⋮	⋮	⋱

FIG. 13 – The table $input.cust$

	kate		...
input items	jan	feb	...
wine	1	1	...
beer	1	0	...
⋮	⋮	⋮	⋱

FIG. 14 – The table $input.items$

wants to compare the behavior on each item, he/she can use the following query to get the binary table of Figure 14 from that of Figure 13 :

$$input.items(I, C.M) : V \longleftarrow input.cust(C, M.I) : V, M \neq age.$$

Navigation through hierarchy, (roll-up and drill-down) can be used to select the level of detail to construct the binary table used as input for the extraction. For example, suppose the user is only interested in customer who purchased more than 50 drinks during the first quarter :

$$input.cust2(C, M.drinks) : 1 \longleftarrow X = sum(sales(C, firstquarter, drinks)), X > 50, sales(C, \_, \_) : \langle \_ \rangle.$$
$$input.cust2(C, M.drinks) : 0 \longleftarrow X = sum(sales(C, firstquarter, drinks)), X \leq 50, sales(C, \_, \_) : \langle \_ \rangle.$$

Now the user is interested in the full list of categories, whatever the sales :

$$input.cust3(C, I) : 1 \longleftarrow X = sum(sales(C, firstquarter, I)), X \neq 0, in(I, product), sales(C, \_, \_) : \langle \_ \rangle.$$

Roll-ups can also be used to simply filter unwanted attributes. For example, the following rule discards regions where less than 100 items have been sold :

$$input.cust4(C, M.I) : 1 \longleftarrow X = sum(sales(R, M, I)),$$

frequents cust	age		jan		...	support
	lt 50	geq 50	wine	beer	...	
f_1	1	0	1	1	...	0.1
f_2	1	0	1	0	...	0.03
⋮	⋮	⋮	⋮	⋮	⋱	

FIG. 15 – The table $frequents\ cust$

frequents items	kate		...
	jan	feb	...
f_1	1	1	...
f_2	1	0	...
⋮	⋮	⋮	⋱

FIG. 16 – The table $frequents\ items$

$$X > 100, in(R, location),$$
$$in(C, R), sales(C, \_, \_) : \langle \_ \rangle.$$

Once binary tables are built, it is possible to run the mining algorithm. The user may specify support thresholds, or more generally interestingness measures that fit to the mining task. The result can also be modeled as tables and cubes.

4.3 Post-processing queries

Frequent sets can be represented in a binary table where each row name is a frequent set identifier and each column name is an attribute. The cell identified by a given column name (an attribute) and a given row name (a frequent set identifier) contains 1 if this attribute is part of this frequent set. Figure 15 (resp. 16) shows the result of the extraction for the table of Figure 13 (resp. 14).

When it is clear from the context, we use the "generic" table name $frequents$ instead of using $frequents.items$, etc. The same idea applies for tables named $input$ and $rules$.

Presenting the result The representation of frequent item sets introduced above may seem unfriendly, but here again OLAP queries can be used to switch to a convenient presentation. For example, a split query can be used to get a frequent set per table :

$$F(M, I) : 1 \longleftarrow frequents(F, M.I) : 1, M \neq age.$$

Note that the age attribute can be taken apart form the other attributes since for a particular frequent item set, its value holds for every combination of the other attributes.

$$F(age, value) : A \longleftarrow frequents(F, age.A) : 1.$$

Useful queries The representation of the frequent sets in a binary table provides an easy way to write some typical post-processing queries. For example, inclusion of frequent sets can be simply written :

$$frequentsInclusion(F1, F2) \longleftarrow$$
$$not(nonInclude(F1, F2)),$$
$$frequentId(F1),$$
$$frequentId(F2).$$
$$nonInclude(F1, F2) \longleftarrow$$
$$frequents(F2, A.B) : 0,$$
$$frequents(F1, A.B) : 1.$$

Where *frequent-id* range over the frequent set identifiers.

Item set complement can also be defined by means of a ternary predicate $complement(F1, F2, F3)$ which holds if $F1 = F2 \cup F3$ and $F2 \cap F3 = \emptyset$.

$$complement(F1, F2, F3) \longleftarrow frequentId(F1),$$
$$frequentId(F2),$$
$$frequentId(F3),$$
$$frequentsInclusion(F2, F1),$$
$$frequentsInclusion(F3, F1),$$
$$not(intersect(F3, F2)),$$
$$not(uncovered(F1, F2, F3)).$$

where *uncovered(F1,F2,F3)* holds if $F1 \subseteq F2 \cup F3$, and is defined by the folowing rule :

$$uncovered(F1, F2, F3) \longleftarrow frequents(F1, A.B) : 1,$$
$$frequents(F2, A.B) : 0,$$
$$frequents(F3, A.B) : 0.$$

and *intersect(F2,F3)* is straightforwardly defined by :

$$intersect(F1, F2) \longleftarrow frequent(F2, A.B) : 1,$$
$$frequent(F3, A.B) : 1.$$

Support, confidence, interestingness measures Association rules and their interestingness measures can be derived from the $frequents$ table by queries. Let us compute the support of the rules of the form $F2 \Rightarrow F3$ as follows.

$$ruleSupport(F2, F3) : S \longleftarrow$$
$$frequents(F1, support) : S,$$
$$complement(F1, F2, F3).$$

Then rule confidence can be computed by :

$$ruleConf(F2, F3) : C \longleftarrow$$
$$frequents(F1, support) : S1,$$
$$frequents(F2, support) : S2,$$
$$complement(F1, F2, F3),$$
$$C = S1/S2.$$

Other interestingness measures can be specified. For example, conviction [10] can be expressed by :

$$ruleConvic(F2, F3) : C \longleftarrow complement(F1, F2, F3),$$
$$frequents(F1, support) : S1,$$
$$frequents(F2, support) : S2,$$
$$frequents(F3, support) : S3,$$
$$C = (S2 \times (1 - S3))/(S2 - S1).$$

Presenting the association rules A table $rules$ can be constructed from the tables *ruleConf* and *ruleSupport*, such that each cell of the table describes a rule :

$$rules(F1, F2) : \langle C, S \rangle \longleftarrow ruleSupport(F1, F2) : S,$$
$$ruleConf(F1, F2) : C$$

rules	f_1	f_2	...
f_1	C_{11} S_{11}	C_{12} S_{12}	...
f_2	C_{21} S_{21}	C_{22} S_{22}	...
⋮	⋮	⋮	⋮

FIG. 17 – The table $rules$

Selection/filtering of rules can now be expressed on the table $rules$. For instance, the following query asks for the rules having confidence higher than 90 % :

$$? - rules(F1, F2) : \langle C, S \rangle, C \geq 0.9$$

The following query asks for the rules whose head contains $beer$:

$?- rules(F1, F2) : \langle C, S\rangle, frequents.cust(F2, \_.beer).$

This one asks for pairs of rules such that the body of the first and the head of the second share an item :

$$?- rules(F1, F2) : \langle C1, S1\rangle, rules(F3, F4) : \langle C2, S2\rangle,$$
$$frequents.cust(F1, A.B),$$
$$frequents.cust(F4, A.B).$$

Links to raw data Having a unique data model enables that raw data and mined patterns are uniformly accessed. For instance, assume the user wants to get the customers violating a given rule, this can be obtained by the following rules :

$$violates(C, F1.F2) \longleftarrow rules(F1, F2) : \langle \_, \_\rangle,$$
$$inputInclusion(F1, C),$$
$$not(inputInclusion(F2, C)).$$
$$inputInclusion(F, C) \longleftarrow frequentId(F),$$
$$input.cust(C, \_.\_) : 1,$$
$$not(nonIncludeInput(F, C)).$$
$$nonIncludeInput(F, C) \longleftarrow$$
$$frequents.cust(F, A.B) : 1,$$
$$input.cust(C, A.B) : 0.$$

We can write generic queries by using variables in table names. For instance, it is straightforward to build tables $violates.X(C, F1.F2)$, $inputInclusion.X(F, C)$ and $nonIncludeInput.X(F, C)$ where X will be bound to $cust$ to focus on customers or $items$ to deal with items.

Rolling-up and drilling down patterns The frequent sets can be hierarchized in the same way than the other dimensions are hierarchized in the OLAP framework. A natural hierarchy for the frequent is the frequent lattice, that can be set up by using the inclusion query :

$$in(F1, F2) \longleftarrow frequentInclusion(F1, F2).$$

In that case, rolling up could mean dropping an attribute from a frequent itemset, whereas drilling down could mean adding an item to a frequent itemset.

The language provides an easy way to define the transitive closure of the in relation :

$$inTrans(X, Y) \longleftarrow in(X, Y).$$
$$inTrans(X, Z) \longleftarrow in(X, Y), inTrans(Y, Z).$$

As an example, from rule $F1 \Rightarrow F2$, we can find the rules having the same body but a larger head :

$$?- rules(F1, F2) : \langle \_, \_\rangle, rules(F1, F3) : \langle \_, \_\rangle,$$
$$inTrans(F2, F3).$$

The in relation can also be used to represent how items are grouped to compose a frequent itemset :

$$in(A.B, F) \longleftarrow frequent(F, A.B) : 1.$$

Using this hierarchy allows to represent rules under a multidimensional form that displays attributes (items). Suppose the user is interested in rules with 2 attributes in the body and 1 attribute in the head. With the following query, such rules can be represented as a table where the attributes in the body are represented as row names and the attribute in the head as column names :

$$rulesXYZ(X.Y, Z) : \langle C, S\rangle \longleftarrow rules(F1, F2) : \langle C, S\rangle$$
$$2 = count(frequents(F1, F1)),$$
$$frequents(F1, \_.X) : 1,$$
$$frequents(F1, \_.Y) : 1,$$
$$1 = count(frequents(F2, F2)),$$
$$frequents(F2, \_.Z) : 1.$$

rulesXYZ		age lt 50	age geq 50	...
wine	beer	C_{11} S_{11}	C_{12} S_{12}	...
	bread	C_{21} S_{21}	C_{22} S_{22}	...
	⋮	⋮	⋮	⋮

FIG. 18 – The table $rulesXYZ$

This multidimensional representation can be manipulated with OLAP operations, by e.g., unesting the row names or spliting according to the column names. It can also be the basis for statistics. Consider the table of Figure 18. Suppose the user is interested in knowing the average confidence of the rules contained in this table. This can be expressed with the following program. First, discard the support

$$rulesConf(A1.A2, A3) : \langle C\rangle \longleftarrow$$
$$rulesXYZ(A1.A2, A3) : \langle C, S\rangle.$$

Then create an attribute representing the grouping of every attribute occuring in the table :

$$in(A, all) \longleftarrow rulesConf(A.\_, \_) : \langle \_\rangle.$$
$$in(A, all) \longleftarrow rulesConf(\_.A, \_) : \langle \_\rangle.$$
$$in(A, all) \longleftarrow rulesConf(\_.\_, A) : \langle \_\rangle.$$

Finally, compute the statistics by aggregating the whole table :

$$avgConfXYZ(X) \longleftarrow$$
$$X = average(ruleConf(all.all, all)).$$

Templates Templates have been introduced in [24] as conditions (1) on the size of the rules, and (2) on the occurrence of attributes in their head or body.

In the first case, suppose the user wants to find the rules having at least 3 attributes in their body :

$$?- rules(F1, F2) : \langle C, S\rangle,$$
$$N = count(frequent(F1, F1)), N > 3,$$
$$frequentId(F1), frequentId(F2).$$

For (2) now assume the user wants to select the rules whose body concern customers having a high income (on the frequent sets mined on the second binary table). This condition can be expressed with the following rule :

$$get(F1, F2) : \langle C, S\rangle \longleftarrow rules(F1, F2) : \langle C, S\rangle,$$
$$frequents.items(F1, Cust.\_) : 1,$$
$$cust(Cust, income) : I,$$
$$I \geq 2000.$$

Finding ancestors In the context of multilevel data, [32] defines the ancestors of a frequent itemset to be the frequent itemsets having the same number of items where one (or more) item is replaced with its ancestor. Ancestors can be computed by the following rule :

$$frequentAncestor(F2, F1) \longleftarrow frequentId(F1),$$
$$frequentId(F2),$$
$$F1 \neq F2,$$
$$C1 = count(frequent(F1, F1)),$$
$$C2 = count(frequent(F2, F2)),$$
$$C1 = C2,$$

$$not(higherLevel(F1, F2)).$$

where ***higherLevel*** is defined as :

$$higherLevel(F1, F2) \longleftarrow frequent(F1, A) : 1,\ frequent(F2, B) : 1,\ in(B, A).$$

[32] defines the ancestors of a rule to be the rules where frequent ancestors appear in the body and/or in the head. They can be found by the program :

$$ruleAncestors(F1.F2, F3.F2) \longleftarrow rule(F1, F2) : \langle \_, \_ \rangle, rule(F3, F2) : \langle \_, \_ \rangle, frequentAncestor(F1, F3).$$

$$ruleAncestors(F1.F2, F1.F3) \longleftarrow rule(F1, F2) : \langle \_, \_ \rangle, rule(F1, F3) : \langle \_, \_ \rangle, frequentAncestor(F2, F3).$$

$$ruleAncestors(F1.F2, F3.F4) \longleftarrow rule(F1, F2) : \langle \_, \_ \rangle, rule(F3, F4) : \langle \_, \_ \rangle, frequentAncestor(F1, F3), frequentAncestor(F2, F4).$$

According to [32], a rule is interesting if it has no ancestors. Such interesting rules can be found with the following query :

$$?-rules(F1, F2) : \langle \_, \_ \rangle, not(ruleAncestors(F1.F2, \_.\_)).$$

Covering Rules can be compared with each other. For example, structural covering [33] consists in filtering the rules being less general than a particular one. A rule $R1$ is less general than a rule $R2$ if the confidence of $R1$ is lesser than the confidence of $R2$, and $R1$ has a shorter head and a longer body than $R2$. Computing covers of discovered association rules is clearly needed in order to remove some redundant rules. This is semantically clear for logical rules (rules with confidence 1). It needs some careful interpretation with rules whose confidence is less than 1.

$$covers(F1.F2, F3.F4) \longleftarrow rules(F1, F2) : \langle C1, S1 \rangle, rules(F3, F4) : \langle C2, S2 \rangle, C1 \geq C2, frequentInclusion(F1, F3), frequentInclusion(F4, F2).$$

Such a rule is quite close to the specification of the task. Obviously, it is one of the major advantage of a rule-based language with a formal declarative semantics. It is clear that evaluating such queries can be very difficult even though operational semantics have been already studied.

The data model and the query language can be used to enhance the different steps of the KDD process. Our aim is not to act on the mining algorithm. Instead we can act on the different tables or cubes used during the process : the $input.X$ binary table, the $frequents.X$ table that provides the frequent item sets and the $rules.X$ table that provides rules. Different mining phases can be performed for more or less complex post-processing phases (e.g., querying simultaneously $frequents.cust$ and $frequents.items$ tables).

5 Conclusion

This paper has demonstrated that an OLAP manipulation language enables to express typical pre-processing and post-processing within the association rule mining context. The uniform access to data and patterns is interesting : it means that the whole process (beside the mining phases) is a collection of queries written with the same language. [16] presents a similar approach by using OQL as a query language. Indeed, in that case, multidimensional features are not considered. Dedicated languages like for instance MINE RULE [27] or MSQL [23] also enable to select data and specify various mining tasks. However, they do not consider the multidimensional aspect of data and patterns.

Modeling KDD processes as queries give rise to optimization possibilities e.g., when complex processes have to be reused on various datasets. This research is part of a project related to the inductive database framework [25, 7, 15]. An inductive database is a database that contains intensionally defined generalizations about the data, in addition to the usual data. The KDD process can then be described as a sequence of queries on an inductive database and mining phases are part of the evaluation process for queries that select specific inductive properties of the data. Indeed, implementing inductive databases, not only for association rules but also for other classes of patterns, will be a long term effort.

We consider that studying typical queries using languages with a clean formal semantics is one mean to understand interactivity during KDD processes. The next step of such a research consists in relaxing the assumption that condensed representation of data (e.g., frequent sets) are materialized (i.e., computed beforehand by some data mining algorithm) but instead that computing it is part of the query evaluation process.

Références

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*. Reading, 1995. 685p.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, Washington, D.C., USA, May 1993. ACM.

[3] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[4] E. F. Codd Associates. Providing OLAP (On-Line Analytical Processing) to user-analysts : An IT mandate [on-line]. 31p. White Paper Available from <http ://www.hyperion.com/solutions.cfm>, 1993.

[5] Elena Baralis and Giuseppe Psaila. Designing templates for mining association rules. *Journal of Intelligent Information Systems*, 9(1) :7 – 32, January 1997.

[6] Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings of the Fourth Pacif-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, volume 1805 of *Lecture Notes in Artificial Intelligence*, pages 62 – 73, Kyoto, JP, April 2000. Springer-Verlag.

[7] Jean-François Boulicaut, Mika Klemettinen, and Heikki Mannila. Modeling KDD processes within the inductive database framework. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, volume 1676, pages 293 – 302, Florence, I, September 1999. Springer-Verlag.

[8] Jean-François Boulicaut, Patrick Marcel, and Christophe Rigotti. Knowledge discovery in multidimensional data. In *Proceedings of the ACM Second International Workshop on Data Warehousing and On-Line Analytical Processing DOLAP'99*, pages 87 – 93, Kansas City, Missouri, USA, November 1999. ACM Press.

[9] J.F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proc. of the 4th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, pages 75–85, Lyon, France, September 2000.

[10] Sergey Brin, Rejeev Motwani, and Craig Silverstein. Beyond market baskets : Generalizing association rules to correlations. In *Proc. of ACM SIGMOD Conf. on Management of Data (SIGMOD'97)*, pages 265 – 276, Tucson, AZ, May 1997. ACM.

[11] A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proc. of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2001)*, pages 267–273, Santa Barbara, CA, USA, 2001.

[12] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD RECORDS*, 26(1) :65–74, March 1997.

[13] Weidong Chen, Michael Kifer, and David Scott Warren. HiLog : a foundation for higher-order logic programming. 15(3) :187–230, February 1993.

[14] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[15] Fosca Giannotti and Giuseppe Manco. Querying inductive databases via logic-based user defined aggregates. In *Proceedings of the Third European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, volume 1704 of *Lecture Notes in Artificial Intelligence*, pages 125–135, Praha, CZ, 1999. Springer-Verlag.

[16] Bart Goethals, Jan Van den Bussche, and Koen Vanhoof. Decision support queries for the interpretation of data mining results. Manuscript, University of Limburg (Belgium), 1998. Available at http ://www.luc.ac.be/∼vdbuss.

[17] Sylvie Guillaume, Fabrice Guillet, and Jacques Philippé. Improving the discovery of association rules with intensity of implication (short paper). In *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, volume 1510 of *Lecture Notes in Artificial Intelligence*, pages 318–327, Nantes, F, 1998. Springer-Verlag.

[18] Mohand-Said Hacid, Patrick Marcel, and Christophe Rigotti. A rule-based data manipulation language for OLAP systems (short paper). In *Proc. Deductive and Object Oriented Databases DOOD'97*, volume 1341, pages 417–418, Montreux (CH), December 1997. Springer-Verlag.

[19] Mohand-Said Hacid, Patrick Marcel, and Christophe Rigotti. A rule-based language for ordered multidimensional databases. In *Proc. 5th Workshop on Deductive Database and Logic Programming (DDLP'97)*, volume 317 of *GMD-Studien*, pages 69–81, Leuven (B), July 1997.

[20] Jiawei Han. OLAP mining : An integration of OLAP with data mining. In *Proc. 1997 IFIP Conference on Data Semantics (DS-7)*, pages 1–11, Leysin (CH), October 1997.

[21] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 1–12, Dallas, Texas, May 2000.

[22] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11) :58 – 64, November 1996.

[23] Tomasz Imielinski and Aashu Virmani. MSQL : A query language for database mining. *Data Mining and Knowledge Discovery*, 3(4) :373 – 408, 1999.

[24] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM'94)*, pages 401 – 407, Gaithersburg, MD, USA, November 1994. ACM.

[25] Heikki Mannila. Inductive databases and condensed representations for data mining. In *Proc. of the Int. Logic Programming Symposium (ILPS'97)*, pages 21 – 30, Port Jefferson, USA, October 13-16 1997. MIT Press.

[26] Patrick Marcel. Modeling and querying multidimensional databases : An overview. *Networking and Information System Journal*, 2(5/6) :515–548, 1999.

[27] Rosa Meo, Giuseppe Psaila, and Stefano Ceri. A new SQL-like operator for mining association rules. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 122 – 133, Mumbay, India, September 1996. Morgan Kaufmann.

[28] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1) :25 – 46, January 1999.

[29] Jian Pei, Jiawei Han, and Runying Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'00)*, Dallas, USA, May 2000.

[30] Arie Shoshani. Olap and statistical databases : Similarities and differences. pages 185–196, Tucson, 1997. ACM.

[31] Padhraic Smyth and Rodney M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4) :301 – 316, August 1992.

[32] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 407 – 419, Zürich, Switzerland, September 1995. Morgan Kaufmann.

[33] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen, and Heikki Mannila. Pruning and grouping of discovered association rules. In *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pages 47 – 52, Heraklion, Crete, Greece, April 1995. MLnet.