Mining Frequent Queries in Star Schemes

Tao-Yuan Jen*, Dominique Laurent*
Nicolas Spyratos**, Oumar Sy***
*LICP, Université de Cergy-Pontoise, 95302 Cergy-Pontoise Cedex, FRANCE
{tao-yuan.jen,dominique.laurent}@dept-info.u-cergy.fr
**LRI, Université Paris 11, 91405 Orsay Cedex, FRANCE
spyratos@lri.fr
***Université Gaston Berger, Saint-Louis, SENEGAL
oumar.sy@ugb.sn

Résumé. L'extraction de *toutes* les requêtes fréquentes dans une base de données relationnelle est un problème difficile, même si l'on ne considère que des requêtes conjonctives. Nous montrons que ce problème devient possible dans le cas suivant : le schéma de la base est un schéma en étoile, et les données satisfont un ensemble de dépendances fonctionnelles et de contraintes référentielles. De plus, les schémas en étoile sont appropriés pour les entrepôts de données et que les dépendances fonctionnelles et les contraintes référentielles sont les contraintes les plus usuelles dans les bases de données. En considérant le modèle des instances faibles, nous montrons que les requêtes fréquentes exprimées par sélection-projection peuvent être extraites par des algorithmes de type Apriori.

1 Introduction

The general problem of mining *all* frequent queries in a (relational) database, *i.e.,* all queries whose answer has a cardinality above a given threshold, is known to be intractable, even if we consider conjunctive queries only (Goethals 2004).

However, mining all frequent queries from a database allows for the production of relevant association rules that cannot be obtained by other approaches, even when dealing with multiple tables, such as in (Dehaspe and Raedt 1997; Diop *et al.* 2002; Faye *et al.* 1999; Han *et al.* 1996; Meo *et al.* 1997; Turmeaux *et al.* 2003). This is so because, in these approaches, association rules are mined in the *same* table. On the other hand, when mining all frequent queries, it is possible to obtain rules whose left and right hand sides are frequent queries mined in *different* tables. The following example, that serves as a running example throughout the paper, illustrates this point.

Example 1 *Let Δ be a database containing three tables, Cust, Prod and Sales, dealing with customers, products and sales transactions, respectively, and suppose that :*

- *the table Cust is defined over the attributes Cid, Cname and Caddr, standing respectively for the identifiers, the names and the addresses of customers,*
- *the table Prod is defined over the attributes Pid and Ptype, standing respectively for the identifiers and the types of products,*
- *the table Sales is defined over the attributes Cid, Pid and Qty where Qty stands for the quantity of a product bought by a customer.*

If all frequent queries in Δ *can be mined, then it is possible to mine a rule such as :* At least 80% of customers living in Paris buy beer. *Indeed, this is stated by the facts that the queries* $q_1 = \pi_{Cid}(\sigma_{Caddr=Paris}(Cust))$ *and* $q_2 = \pi_{Cid}(\sigma_{Caddr=Paris \wedge Ptype=beer}(Cust \bowtie Prod \bowtie Sales))$ *are frequent, and that the confidence of the rule* $q_1 \Rightarrow q_2$ *is greater than or equal to* 80%.

Notice that, in q_1, *Cust cannot be replaced with* $Cust \bowtie Prod \bowtie Sales$ *because, in this case, customers with no transactions would not be taken into account.* □

In this paper, we show that the problem of mining all frequent queries in a database becomes tractable under restrictions that are met in data warehousing. Indeed, data warehouses are organized according to *star schemes,* over which constraints such as functional dependencies and referential constraints are assumed. The main contribution of this paper is to show that for a database over a star scheme, all selection-projection queries that are frequent can be computed based on any level-wise algorithm such as Apriori (Agrawal *et al.* 1996), and thus, that this computation is tractable.

In our formalism, a database Δ satisfying a set FD of functional dependencies is represented by its *weak instance*, denoted by Δ_{FD} (Ullman 1988). Roughly speaking, Δ_{FD} is a set of tuples over the set U of all attributes, and the difference between tuples in Δ_{FD} and standard tuples is that tuples in Δ_{FD} may contain null values.

Using this unique table Δ_{FD}, we consider all queries of the form $\sigma_S(\pi^{\downarrow}_X(\Delta_{FD}))$, where S is a conjunctive selection condition, X is any set of attributes and $\pi^{\downarrow}_X(\Delta_{FD})$ is the *total* projection of Δ_{FD} over X (*i.e.,* all restrictions over X of tuples in Δ_{FD} whose values over X are constants). Given such a query q, the *answer to* q *in* Δ, denoted by $ans_\Delta(q)$, is the set of all tuples in $\pi^{\downarrow}_X(\Delta_{FD})$ that satisfy the selection condition S. Then, the *support of* q *in* Δ, denoted by $sup_\Delta(q)$, is the cardinality of $ans_\Delta(q)$.

Example 2 *Referring back to Example 1, assume that Cust satisfies the functional dependencies* $Cid \rightarrow Cname$ *and* $Cid \rightarrow Caddr$, *that Prod satisfies the functional dependency* $Pid \rightarrow Ptype$ *and that Sales satisfies the functional dependency* $Cid, Pid \rightarrow Qty$. *Then the scheme of* Δ *is a star scheme in which the fact table is Sales and the two dimensional tables are Cust and Prod.*

Denoting by FD the set of the functional dependencies given above, instead of considering the queries q_1 *and* q_2 *of Example 1, we consider the following queries*

- $q'_1 = \sigma_{Caddr=Paris}(\pi^{\downarrow}_{Cid,Caddr}(\Delta_{FD}))$, *and*
- $q'_2 = \sigma_{Caddr=Paris \wedge Ptype=beer}(\pi^{\downarrow}_{Cid,Caddr,Ptype}(\Delta_{FD}))$.

We note that, although q'_1 *and* q'_2 *involve the same table* Δ_{FD}, *the computation of the supports (and thus of the confidence) takes into account the fact that there may exist customers in the table Cust whose identifiers do not occur in the table Sales.* □

We show that if the scheme of Δ is a star scheme, then the problem of mining all frequent conjunctive queries can be treated according to the following two steps, each of them being based on a level-wise algorithm such as Apriori (Agrawal *et al.* 1996) :

- Step 1 : compute all frequent queries of the form $\pi^{\downarrow}_X(\Delta_{FD})$.
- Step 2 : for each relation scheme X such that $\pi^{\downarrow}_X(\Delta_{FD})$ is frequent, compute all frequent queries of the form $\sigma_S(\pi^{\downarrow}_X(\Delta_{FD}))$, where S is a conjunction of selection conditions of the form $A = a$ with A in X and a in $dom(A)$.

To our knowledge, except in (Goethals 2004), no other work addresses the general problem of computing all frequent queries in a given database. The work in (Goethals 2004) considers *conjunctive* queries, as we do in this paper and points out that considering no restrictions on the database scheme and no functional dependencies leads to a non tractable complexity. Although some hints on possible restrictions are mentioned in (Goethals 2004), no specific case is actually studied. As we shall see, considering a star scheme with its associated constraints leads to tractable level-wise algorithms.

The approach of (Casali *et al.* 2003) is also related to our work because data cubes and star schemes both deal with multi dimensional data. However, the frequent queries considered in (Casali *et al.* 2003) involve the fact table only. Therefore, contrary to our approach, it is not possible to mine frequent queries defined on any set of attributes. We note however that (Casali *et al.* 2003) takes into account the hierarchies on the dimensions, which is not the case in our approach.

As mentioned previously, all approaches dealing with mining frequent queries in multi-relational databases (Dehaspe and Raedt 1997; Diop *et al.* 2002; Faye *et al.* 1999; Han *et al.* 1996; Meo *et al.* 1997; Turmeaux *et al.* 2003) consider only one table for a given mining task, and consequently, these approaches fail to mine association rules as in Example 1. We also note that, except for (Turmeaux *et al.* 2003), all these approaches are restricted to conjunctive queries, as we do in this paper.

The paper is organized as follows : In Section 2, we recall the basics of weak instance semantics and of star schemes. Section 3 deals with the queries that are of interest in our approach. In Section 4, two algorithms are provided for the computation of frequent queries and Section 5 concludes the paper.

2 Background

We recall from (Ullman 1988) that, in the relational model of databases, given a universe of attributes U, every attribute A in U is associated with a set of values called the domain of A, and denoted by $dom(A)$. Moreover, a relational database scheme consists of a set $\mathcal{S}$ of tables $\tau_1, \ldots, \tau_n$, where each table τ_i is associated with an attribute set, called the scheme of τ_i and denoted by $sch(\tau_i)$. In a relational database over $\mathcal{S}$ each table τ_i contains a relation over $sch(\tau_i)$, *i.e.,* a finite set of tuples over $sch(\tau_i)$.

2.1 Universal Relation Scheme Interfaces

Universal relation scheme interfaces were introduced in the early 80s in order to provide logical independence to the relational model. Given a database Δ over a universe of attributes U and a set of functional dependencies FD over U, logical independence is achieved by associating (Δ, FD) to a *single* table over U, that we denote by Δ_{FD}, and that is called the *weak instance* of Δ. We refer to (Laurent *et al.* 2003; Ullman 1988) for more details on the contruction of Δ_{FD}, and we simply note here that, contrary to standard relations, tuples in Δ_{FD} may contain *null values.*

The weak instance Δ_{FD} can be seen as the only table to which queries on Δ are addressed. More precisely, for every relation scheme X, we denote by $\pi^{\downarrow}_{X}(\Delta_{FD})$ the set of all tuples t over X such that (i) t contains no null value, and (ii) there exists a

tuple t' in Δ_{FD} such that $t'.A = t.A$ for every attribute A in X. $\pi^{\downarrow}_X(\Delta_{FD})$ is then the set of all tuples over X that are *true* in Δ, and so, it is possible to consider all queries of the form $\sigma_S(\pi^{\downarrow}_X(\Delta_{FD}))$ where S is a selection condition involving attributes in X.

In this paper, we consider only queries whose selection condition is a *conjunction* of selection conditions of the form $A = a$ where A is an attribute in X and a is a constant in $dom(A)$.

2.2 Star Schemes

An *N-dimensional star scheme* consists of a distinguished table φ, called the *fact table*, and N other tables $\delta_1, \ldots, \delta_N$ called *dimension tables*, such that :

1. If $K_1, \ldots, K_N$ are the (primary) keys of $\delta_1, \ldots, \delta_N$, respectively, then $K = K_1 \cup \ldots \cup K_N$ is the key of φ ;
2. For every $i = 1, \ldots, N$, $\pi_{K_i}(\varphi) \subseteq \pi_{K_i}(\delta_i)$ (Note that each K_i is a foreign key in the fact table φ).

The attribute set $M = sch(\varphi) \backslash K$ is called the *measure* of the star scheme. Moreover, we use the following simplified notation : $sch(\delta_1), \ldots, sch(\delta_N)$ are denoted by $D_1, \ldots, D_N$, respectively, and $sch(\varphi)$ is denoted by F.

Example 3 *The scheme of the database (Δ, FD) in our running example is a 2-dimensional star scheme $\{\delta_1, \delta_2, \varphi\}$ where the two dimension tables are $\delta_1 = Cust$ and $\delta_2 = Prod$, the fact table is $\varphi = Sales$ and the measure is $\{Qty\}$. Moreover :*
- *Cid is the key of δ_1, Pid is the key of δ_2, and $Cid\,Pid$ is the key of φ,*
- *$\pi_{Cid}(\varphi) \subseteq \pi_{Cid}(\delta_1)$ and $\pi_{Pid}(\varphi) \subseteq \pi_{Pid}(\delta_2)$.* □

Now, let (Δ, FD) be a database defined over an N-dimensional star scheme and let us consider its weak instance Δ_{FD}.

We "simplify" the table Δ_{FD} by removing from it all tuples t' for which there exists t in Δ_{FD} such that $t'.A = t.A$ for every attribute A over which t' is a constant. We feel justified in performing this simplification because doing so does not change the answers to queries. From now on, the symbol Δ_{FD} will denote the simplified table. It is important to note that the simplified table contains two kinds of tuples :
- either *total* tuples, *i.e.,* tuples containing no null value, and there is a one-to-one mapping between these tuples and the tuples of the fact table,
- or tuples t containing constants over the attributes of a single dimension, say i, such that the key value $t.K_i$ does not occur in the fact table.

We denote by Δ^{φ}_{FD} the set of all total tuples in Δ_{FD} and by Δ^{i}_{FD} the set of all tuples in Δ_{FD} containing constants only over attributes of dimension i, for $i = 1, \ldots, N$.

In the remainder of this paper, we consider a *fixed* N-dimensional star scheme, with fact table φ and dimension tables $\delta_1, \ldots, \delta_N$, and a *fixed* database Δ over that scheme. Moreover, for the sake of simplification, we assume that for every $i = 1, \ldots, N$, the key of dimension i is reduced to a single attribute K_i.

On the other hand, it is well-known that in practice, the cardinality of the fact table is much higher than that of any dimension table. In order to take this situation into account, we assume that for every $i = 1, \ldots, N$, $|\delta_i| \leq |\varphi|$.

Example 4 *We refer back to the database introduced in Example 1, where the universe of attributes is* $U = \{Cid, Cname, Caddr, Pid, Ptype, Qty\}$ *and the functional dependencies are* $FD = \{Cid \rightarrow Cname,\ Cid \rightarrow Caddr,\ Pid \rightarrow Ptype,\ CidPid \rightarrow Qty\}$.

Let us consider a database (Δ, FD) *consisting of the following three relations :*

Cust	*Cid*	*Cname*	*Caddr*
	c_1	*John*	*Paris*
	c_2	*Mary*	*Paris*
	c_3	*Jane*	*Paris*
	c_4	*Anne*	*Tours*

Prod	*Pid*	*Ptype*
	p_1	*milk*
	p_2	*beer*

Sales	*Cid*	*Pid*	*Qty*
	c_1	p_1	10
	c_2	p_2	5
	c_2	p_1	1
	c_1	p_2	10

The simplified weak instance Δ_{FD} *(in which null values are not represented) is the following :*

Δ_{FD}	*Cid*	*Pid*	*Cname*	*Caddr*	*Ptype*	*Qty*
	c_3		*Jane*	*Paris*		
	c_4		*Anne*	*Tours*		
	c_1	p_1	*John*	*Paris*	*milk*	10
	c_2	p_2	*Mary*	*Paris*	*beer*	5
	c_2	p_1	*Mary*	*Paris*	*milk*	1
	c_1	p_2	*John*	*Paris*	*beer*	10

Notice that in this example, Δ^{φ}_{FD} *and* Δ^{1}_{FD} *contain respectively the last four tuples and the first two tuples of* Δ_{FD}*, whereas* Δ^{2}_{FD} *is empty. Moreover, the answers to queries* q'_1 *and* q'_2 *of Example 2 are shown below :*

$ans(q'_1)$	*Cid*	*Caddr*
	c_1	*Paris*
	c_2	*Paris*
	c_3	*Paris*

$ans(q'_2)$	*Cid*	*Caddr*	*Ptype*
	c_2	*Paris*	*beer*
	c_1	*Paris*	*beer*

□

3 Frequent Queries

3.1 Queries

Definition 1 *Given a database* Δ *over an* N*-dimensional star scheme, let* X *be a relation scheme. Denoting by* $\bot$ *and* $\top$ *the false and true selection conditions, respectively, let* $\Sigma(X)$ *be the following set of conjunctive selection conditions :*

$$\Sigma(X) = \{\bot, \top\} \cup \{\ (A_1 = a_1) \wedge \ldots \wedge (A_k = a_k) \mid (\forall i = 1, \ldots, k)(A_i \in X \text{ and } a_i \in dom(A_i)) \text{ and } (\forall i, j \in \{1, \ldots, k\})(i \neq j \Rightarrow A_i \neq A_j)\}.$$

Selection conditions of $\Sigma(X)$ *are called* selection conditions over X*. Moreover, we denote by* $\mathcal{Q}(X)$ *the set of all queries of the form* $\sigma_S(\pi^{\downarrow}_X(\Delta_{FD}))$ *where* $S \in \Sigma(X)$*, and by* $\mathcal{Q}(\Delta)$ *the union of all sets* $\mathcal{Q}(X)$ *for all relation schemes* X*.*

According to Definition 1, we have $\sigma_{\top}(\pi^{\downarrow}_X(\Delta_{FD})) = \pi^{\downarrow}_X(\Delta_{FD})$ and $\sigma_{\bot}\ (\pi^{\downarrow}_X(\Delta_{FD})) = \emptyset$. In order to simplify the notations, we denote $\sigma_S(\pi^{\downarrow}_X(\Delta_{FD}))$ by $\sigma_S(X)$ with the convention that when $S = \top$, $\pi^{\downarrow}_X(\Delta_{FD})$ is denoted by (X).

We compare queries of $\mathcal{Q}(X)$ using the standard query-containment pre-ordering (Ullman 1988) : for all q_1 and q_2 in $\mathcal{Q}(X)$, q_1 is *contained* in q_2, denoted by $q_1 \sqsubseteq q_2$, if for all databases (over the fixed N-dimensional star scheme), we have $ans(q_1) \subseteq ans(q_2)$.

Let us define the following two operators $\overline{\wedge}$ and $\veebar$ on selection conditions in $\Sigma(X)$: For all S_1 and S_2 in $\Sigma(X)$, define :

- If $S_1 = \bot$ or $S_2 = \bot$ then $S_1 \,\overline{\wedge}\, S_2 = \bot$
  If $S_1 = \top$ (respectively $S_2 = \top$) then $S_1 \,\overline{\wedge}\, S_2 = S_2$ (respectively S_1)
  Otherwise if $S_1 \wedge S_2 \in \Sigma(X)$ then $S_1 \,\overline{\wedge}\, S_2 = S_1 \wedge S_2$ else $S_1 \,\overline{\wedge}\, S_2 = \bot$.
- If $S_1 = \bot$ (respectively $S_2 = \bot$) then $S_1 \veebar S_2 = S_2$ (respectively S_1)
  If $S_1 = \top$ or $S_2 = \top$ then $S_1 \veebar S_2 = \top$
  Otherwise $S_1 \veebar S_2$ is the conjunction of all elementary selection conditions ($A_i = a_i$) that occur in both S_1 and S_2.

It can be seen that for all queries $q_1 = \sigma_{S_1}(X)$ and $q_2 = \sigma_{S_2}(X)$ in $\mathcal{Q}(X)$ we have :

1. $\sigma_{S_1 \overline{\wedge} S_2}(X)$ and $\sigma_{S_1 \veebar S_2}(X)$ are in $\mathcal{Q}(X)$
2. $q_i \sqsubseteq \sigma_{S_1 \veebar S_2}(X)$ and $q_i \sqsupseteq \sigma_{S_1 \overline{\wedge} S_2}(X)$ for $i = 1, 2$
3. $\sigma_{S_1 \veebar S_2}(X) = \min_{\sqsubseteq}\{q \in \mathcal{Q}(X) \mid q_1 \sqsubseteq q \text{ and } q_2 \sqsubseteq q\}$ and
   $\sigma_{S_1 \overline{\wedge} S_2}(X) = \max_{\sqsubseteq}\{q \in \mathcal{Q}(X) \mid q \sqsubseteq q_1 \text{ and } q \sqsubseteq q_2\}$.

Thus, $\langle \mathcal{Q}(X), \sqsubseteq \rangle$ is a lattice. This property is used in our approach to compute the frequent queries of $\mathcal{Q}(X)$, using a level-wise algorithm such as Apriori (Agrawal *et al.* 1996).

3.2 Frequent Queries

Definition 2 *Let Δ be a database over an N-dimensional star scheme. For every query q in $\mathcal{Q}(\Delta)$, the* support of q in Δ, *denoted by $sup_\Delta(q)$ (or by $sup(q)$ when Δ is understood), is the cardinality of the answer to q in Δ.*

A query q is said frequent in Δ *(or* frequent *when Δ in understood) if $sup(q)$ is greater than or equal to a given support threshold σ.*

Referring back to Example 4, it is easy to see that we have $sup(q'_1) = 3$ and $sup(q'_2) = 2$. Thus, for a given support threshold equal to 3, q'_1 is frequent whereas q'_2 is not.

It turns out that our notion of support is monotonic with respect to $\sqsubseteq$, *i.e.,* for all queries q_1 and q_2 in $\mathcal{Q}(\Delta)$: $q_1 \sqsubseteq q_2 \Rightarrow sup(q_1) \leq sup(q_2)$.

Since $q_1 \sqsubseteq q_2$ requires that q_1 and q_2 be over the same scheme, the previous implication can be stated as follows : If $q_1 = \sigma_{S_1}(X)$ and $q_2 = \sigma_{S_2}(X)$ are such that $S_2 = S_1 \,\overline{\wedge}\, S$, and if q_1 is not frequent, then q_2 is not frequent either. Consequently, frequent queries of $\mathcal{Q}(\Delta)$ can be computed using any level-wise algorithm. Morever, notice that if (X) is not frequent then no query of $\mathcal{Q}(X)$ is frequent.

On the other hand, given a standard relation r over relation scheme R (*i.e.,* a finite set of tuples defined on all the attributes of R), for all X and Y such that $X \subseteq Y \subseteq R$, we have $|\pi_X(r)| \leq |\pi_Y(r)|$. However, this property does not hold for partial tuples, meaning that $X \subseteq Y \Rightarrow sup((X)) \leq sup((Y))$ does not hold in our framework.

Indeed, in the table of Example 4, with $X = \{Cid, Caddr\}$ and $Y = \{Cid, Caddr, Qty\}$, we have $X \subseteq Y$, whereas $sup((X)) = 4$ and $sup((Y)) = 3$. The following proposition shows cases of monotonicity.

Proposition 1 *Let X and Y be relation schemes satisfying one of the statements :*
either $(\forall i, j \in \{1, \ldots, N\})(X \not\subseteq D_i$ *and* $Y \not\subseteq D_j)$
or $(\exists i \in \{1, \ldots, N\})(X \subseteq D_i$ *and* $Y \subseteq D_i)$.
For every selection condition S over X : $X \subseteq Y \Rightarrow sup(\sigma_S(X)) \leq sup(\sigma_S(Y))$.

Based on Proposition 1, the frequent queries of the form (X) are computed by considering the lattice of all non-empty subsets of U, starting by U itself. At each level, candidate relation schemes of the same cardinality are pruned as follows :

- If for every $i = 1, \ldots, N$, $X \not\subseteq D_i$, and if, for some attribute A, (XA) has been found not frequent, then (X) cannot be frequent.
- If there exists i in $\{1, \ldots, N\}$ such that $X \subset D_i$, and if, for some A in D_i, (XA) has been found not frequent in the previous step, then (X) cannot be frequent.

It is important to note that, for a given dimension i, the fact that (D_iA) is not frequent does *not* imply that (D_i) is not frequent. Moreover, Proposition 1 and our assumption that $|\varphi| \geq |\delta_i|$ $(i = 1, \ldots, N)$, imply the following corollary.

Corollary 1 *For every query q in $\mathcal{Q}(\Delta)$, we have : $sup(q) \leq sup((U))$.*

We use the functional dependencies for further optimization, based on the following : given a standard relation r over relation scheme R (*i.e.,* a finite set of tuples defined on all the attributes of R), if X and Y are such that r satisfies $X \rightarrow Y$, then $|\pi_X(r)| = |\pi_{XY}(r)|$.

Thus, if X and Y have the same key, then for every selection condition S over X and Y we have $|\sigma_S(X)| = |\sigma_S(Y)|$. Moreover, it can be seen that, in the case of a star scheme, every relation scheme X has one key, denoted by $key(X)$, and defined by :

- if for every $i = 1, \ldots, N$, $K_i \in X$ then $key(X) = K_1 \ldots K_N$
- else $key(X) = X \setminus \{A \in X \mid (\exists i \in \{1, \ldots, N\})(K_iA \subseteq D_i \cap X$ and $A \neq K_i)\}$.

Proposition 2 *Let X and Y be two relation schemes such that $key(X) = key(Y)$. For every selection condition S over X and Y, $sup(\sigma_S(X)) = sup(\sigma_S(Y))$.*

Therefore, applying Proposition 2 in the context of a level-wise algorithm, for every candidate query (X), if there exists an attribute A not in X such that $key(X) = key(XA)$ then (X) is frequent if and only if (XA) has been found frequent. The following example illustrates Proposition 1 and Proposition 2.

Example 5 *Consider the database Δ of Example 4 and a support threshold equal to 3. We recall that the query $q'_2 = \sigma_{Caddr=Paris \wedge Ptype=beer}(X)$ where $X = \{Cid, Caddr, Ptype\}$ is not frequent. Therefore, when computing the frequent queries of $\mathcal{Q}(Y)$, where $Y = \{Caddr, Ptype\}$, by Proposition 1, we know,* without any computation, *that the query $\sigma_{Caddr=Paris \wedge Ptype=beer}(Y)$ is not frequent.*

On the other hand, since $sup((U)) = 4$, (U) is frequent. Using Proposition 2, all queries of the form (X) such that $\{Cid, Pid\} \subseteq X$ are frequent. So, when computing the frequent queries of the form (X) with $|X| = 5$, we know, without any computation, *that the following schemes are frequent with a support equal to 4 : $\{Cid, Pid, Caddr, Ptype, Qty\}$, $\{Cid, Pid, Cname, Ptype, Qty\}$, $\{Cid, Pid, Cname, Caddr, Qty\}$ and $\{Cid, Pid, Cname, Caddr, Ptype\}$.* □

4 Algorithms

4.1 Frequent Queries of the Form (X)

In the algorithm given below, we use the following notations :

- For a given integer k, C_k^{φ} denotes the set of candidate relation schemes of cardinality k that are not subsets of any D_i, C_k^i $(i = 1, \ldots, N)$ denotes the set of candidate relation schemes of cardinality k that are subsets of D_i, and C_k denotes the union of C_k^{φ} and of all C_k^i for $i = 1, \ldots, N$.
- Similarly, for a given integer k, L_k^{φ} denotes the set of frequent queries whose corresponding scheme is of cardinality k and not a subset of any D_i, L_k^i $(i = 1, \ldots, N)$ denotes the set of frequent queries whose corresponding scheme is of cardinality k and is a subset of D_i, and L_k denotes the union of L_k^{φ} and of all L_k^i for $i = 1, \ldots, N$.
- $sup^{\varphi}((X))$ and $sup^i((X))$ $(i = 1, \ldots, N)$ denote the number of distinct tuples over X obtained by scanning the tables Δ_{FD}^{φ} and Δ_{FD}^i, respectively. Thus, given a relation scheme X, if there exists i in $\{1, \ldots, N\}$ such that $X \subseteq D_i$ then $sup((X)) = sup^{\varphi}((X)) + sup^i((X))$, else $sup((X)) = sup^{\varphi}((X))$.

Algorithm 1.
Input : The simplified weak instance Δ_{FD} associated to a database Δ over an N-dimensional star scheme $\{D_1, \ldots, D_N, F\}$. The cardinalities $|\varphi|, |\delta_1|, \ldots, |\delta_N|$ of all tables in Δ. A support threshold σ.
Output : All queries of the form (X) that are frequent in Δ.
Method :

```
if |φ| < σ then //(1) no computation based on Corollary 1
   return ∅
else
   k = |U| ; L_k = {(U)} ; L_k^φ = {(U)}
   for each i = 1, ..., N do L_k^i = ∅ end for each
   while L_k ≠ ∅ or k > min_i(|D_i|) do
      k = k − 1
      //(2) generation of candidates of level k by considering all subsets
      //of cardinality k of all relation schemes in L_{k+1}
      generate C_k from L_{k+1}
      C_k^φ = {X ∈ C_k | (∀i = 1, ..., N)(X ⊈ D_i)}
      //(3) pruning based on Proposition 1
      C_k^φ = C_k^φ \ {X ∈ C_k^φ | (∃A ∈ U)((XA) ∉ L_{k+1}^φ)}
      //(4) pruning based on Proposition 2
      same_sup = {X ∈ C_k^φ | (∃A ∈ U)((XA) ∈ L_{k+1}^φ and key(X) = key(XA))}
      C_k^φ = C_k^φ \ same_sup
      L_k^φ = {(X) | X ∈ same_sup}
      for each i = 1, ..., N do
         C_k^i = ∅
         if k ≤ |D_i| then
            if k = |D_i| then //(5) test whether (D_i) is frequent
               if |δ_i| ≥ σ then L_k^i = {(D_i)} else L_k^i = ∅ end if
            else
               C_k^i = {X ∈ C_k | X ⊂ D_i}
```

```
                //(6) pruning based on Proposition 1
                C_k^i = C_k^i \ {X ∈ C_k^i | (∃A ∈ D_i)((XA) ∉ L_{k+1}^i)}
                //(7) pruning based on Proposition 2
                same_sup = {X ∈ C_k^i | K_i ∈ X and (∃A ∈ D_i)((XA) ∈ L_{k+1}^i)}
                C_k^i = C_k^i \ same_sup
                L_k^i = {(X) | X ∈ same_sup}
            end if
        end if
    end for each
    //(8) computation or pre-computation of the supports : one pass over Δ_FD^φ
    compute sup^φ((X)) for each X ∈ C_k^φ ∪ C_k^1 ∪ ... ∪ C_k^N
    L_k^φ = L_k^φ ∪ {(X) | X ∈ C_k^φ and sup^φ((X)) ≥ σ}
    for each i = 1, ..., N do
        //(9) additional pruning
        C_k^i = C_k^i \ {X ∈ C_k^i | sup^φ((X)) + |δ_i| < σ}
        //(10) computation of the supports : one pass over Δ_FD^i
        L_k^i = L_k^i ∪ {(X) | X ∈ C_k^i and sup^φ((X)) + sup^i((X)) ≥ σ}
    end for each
    L_k = L_k^φ ∪ L_k^1 ∪ ... ∪ L_k^N
  end while
  return ⋃_k(L_k)
end if
```

We give more details on comments in Algorithm 1. Comment `(1)` is an immediate consequence of Corollary 1, and concerning comment `(2)`, we note that all candidates at level k are obtained by removing one attribute from the frequent schemes at level $k+1$. We do not give details about the function `generate`, but we mention that considering the attributes according to a fixed ordering allows to efficiently perform this step.

Comments `(3)` and `(6)` are consequences of Proposition 1. In particular, it is important to note that, in the case of comment `(6)`, C_k^i is pruned by considering subsets of D_i only. Similarly, comments `(4)` and `(7)` are consequences of Proposition 2.

The test in comment `(5)` comes from the fact that, for every $i = 1, \ldots, N$, the support of (D_i) is equal to $|\delta_i|$, and thus requires no computation. Moreover, due to Proposition 1, all dimensions D_i must be considered even if not generated from the previous step. This comment is also related to the condition in the `while` loop.

Comments `(8)` and `(10)` deal with the count of $sup((X))$: if X is not contained in any dimension scheme D_i then $sup((X)) = sup^{\varphi}((X))$, and if X is a subset of some D_i, then $sup((X)) = sup^{\varphi}((X)) + sup^{i}((X))$. In this case, an additional pruning is mentioned in comment `(9)`. Indeed, for every $X \subseteq D_i$, we know that $sup((X)) \leq sup^{\varphi}((X)) + |\delta_i|$. Therefore, if $sup^{\varphi}((X)) + |\delta_i| < \sigma$, then (X) is not frequent.

Summarizing the remarks just above, it can be seen that Algorithm 1 is a level-wise algorithm that scans the table Δ_{FD} only once per level, and that more prunings than in Apriori are possible.

4.2 Frequent Queries of the Form $\sigma_S(X)$

The frequent queries of the form $\sigma_S(X)$ are computed by applying the algorithm Apriori to each frequent query in the output of Algorithm 1. The queries in $\bigcup_k(L_k)$

are considered in a decreasing ordering of the cardinalities of the schemes, in order to prune candidate queries based on Proposition 1 and Proposition 2.

In the algorithm given below, we denote by C_k^X (respectively L_k^X) the set of all candidate (respectively frequent) queries of the form $\sigma_S(X)$ where S is a selection condition over X containing exactly k conjuncts $(A = a)$. Moreover, L^X denotes the set of all queries in $\mathcal{Q}(X)$ that are frequent.

Algorithm 2.
Input : The simplified weak instance table Δ_{FD} associated to a database Δ over an N-dimensional star scheme $\{D_1, \ldots, D_N, F\}$. The output $\bigcup_k(L_k)$ of Algorithm 1 and the corresponding support threshold σ.
Output : All queries of the form $\sigma_S(X)$ that are frequent in Δ.
Method :

`for each` (X) in $\bigcup_k(L_k)$ `do`
  `if` $(\exists A \in U)(key(X) = key(XA))$ `then //(1)` no computation based on Proposition 2
    $L^X = \{\sigma_S(X) \mid \sigma_S(XA) \in L^{XA}$ and S is over $X\}$
  `else`
    $k = 0$; $L_k^X = \{(X)\}$; $S_k^X = \top$
    `while` $L_k^X \neq \emptyset$ and $k < |X|$ `do`
      $k = k + 1$
      `//(2)` generation of all selection conditions based on Definition 1
      `generate` the set S_k^X of all selection conditions over X based on L_{k-1}^X
      $C_k^X = \{\sigma_S(X) \mid S \in S_k^X\}$
      `//(3)` standard pruning
      $C_k^X = C_k^X \setminus \{\sigma_S(X) \mid (\exists S')(S = S' \wedge (A = a)$ and $\sigma_{S'}(X) \notin L_{k-1}^X)\}$
      `//(4)` pruning based on Proposition 1
      `if` $(\forall i = 1, \ldots, N)(X \neq D_i)$ `then`
        $C_k^X = C_k^X \setminus \{\sigma_S(X) \mid (\exists A \in U)(\sigma_S(XA) \notin L^{XA})\}$
      `enf if`
      `//(5)` computation of the supports : if $(\exists i \in \{1, \ldots, N\})(X \subseteq D_i)$
      //then one pass over $\Delta_{FD}^{\varphi} \cup \Delta_{FD}^{i}$ (or δ_i) else one pass over Δ_{FD}^{φ}
      `compute` $sup(\sigma_S(X))$ for each $\sigma_S(X) \in C_k^X$
      $L_k^X = L_k^X \cup \{\sigma_S(X) \mid \sigma_S(X) \in C_k^X$ and $sup(\sigma_S(X)) \geq \sigma\}$
    `end while`
    $L^X = \bigcup_k(L_k^X)$
  `end if`
`end for each`
`return` $L = \bigcup_X(L^X)$

We review the main steps of Algorithm 2 as follows : as stated in comment `(1)`, Proposition 2 implies that L^X can be obtained from L^Y, if $X \subseteq Y$ and $key(X) = key(Y)$. Concerning comment `(2)`, we simply note that selection conditions at level k are obtained from the selection conditions in the queries of L_{k-1}^X in much the same way as in Apriori. Comment `(3)` refers to the standard pruning phase of Apriori, and as noticed in comment `(4)`, Proposition 1 allows for additional pruning, based on the fact that, when computing L^X, if for some superset Y of X, L^Y has already been computed, then we know that for every selection condition S over X, $sup(\sigma_S(X)) \leq sup(\sigma_S(Y))$. Comment `(5)` specifies the part of the table Δ_{FD} that has to be scanned to compute

the supports. We note that if $X \subseteq D_i$ then scanning δ_i could be an option, instead of scanning $\Delta^{\varphi}_{FD} \cup \Delta^{i}_{FD}$.

4.3 Computational Issues

Although we have no experiments to report, we would like to point out some comments on our approach. First, we note that the table Δ_{FD} can be efficiently computed, using for instance external joins (Ullman 1988). On the other hand, finding all frequent queries in a database over an N-dimensional star scheme requires at most $P+1$ applications of a level-wise algorithm, if P schemes X are such that (X) is frequent.

However, it is likely that not all frequent queries are of interest for each user. Instead, some users can be interested in some schemes while other users would like to focuss on other schemes. We propose the following policy :

- Run Algorithm 1 once for all users. Storing all frequent queries of the form (X) with their supports could serve as a basis for queries issued by the users.
- Assuming that users ask for frequent queries on different schemes (but rarely all of them), it is easy to modify Algorithm 2 so that it is restricted to a specified set of schemes, at the cost of less additional prunings.
- If all frequent queries computed so far are stored with their supports, then additional prunings are possible, as done in (Diop *et al.* 2002).

5 Conclusion and Further Work

In this paper, we have considered the weak instance model of relational databases, in order to design level-wise algorithms for the computation of *all* frequent queries in a database over an N-dimensional star scheme. Moreover, we have shown that, in this case, additional prunings with respect to Apriori are possible.

We are currently implementing our approach, and future research directions include the following : (i) Considering schemes more sophisticated than star schemes, such as snowflake or constellation schemes. The work reported in (Levene and Loizou 2003) provides a suitable theoretical basis for this investigation. (ii) Since our work is closely related to the approach presented in (Diop *et al.* 2002), we are investigating the relationships between the two approaches. (iii) Since data cubes and star schemes are two notions that deal with multi dimensional data, the relationships between our work and that of (Casali *et al.* 2003) must be investigated further.

Références

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. (1996). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 309–328. AAAI-MIT Press.

Casali, A., Cichetti, R., and Lakhal, L. (2003). Extracting semantics from data cubes using cube transversals and closures. In *KDD*, pages 69–78. ACM.

Dehaspe, L. and Raedt, L. D. (1997). Mining association rules in multiple relations. In S. Dzeroski and N. Lavrac, editors, *7th International Workshop on Inductive Logic Programming*, volume LNCS 1297, pages 125–132. Springer Verlag.

Diop, C. T., Giacometti, A., Laurent, D., and Spyratos, N. (2002). Composition of mining contexts for efficient extraction of association rules. In *EDBT'02*, volume LNCS 2287, pages 106–123. Springer Verlag.

Faye, A., Giacometti, A., Laurent, D., and Spyratos, N. (1999). Mining rules in databases with multiple tables : Problems and perspectives. In *3rd International Conference on Computing Anticipatory Systems (CASYS'99)*, Liège, Belgique.

Goethals, B. (2004). Mining queries (unpublished paper). In *Workshop on inductive databases and constraint based mining*, Hinterzarten (Germany), available at http ://www.informatik.uni-freiburg.de/˜ml/IDB/talks/Goethals_slides.pdf.

Han, J., Fu, Y., Wang, W., Koperski, K., and Zaiane, O. (1996). Dmql : A data mining query language for relational databases. In *SIGMOD Workshop DMKD'96*, pages 27–34.

Laurent, D., Luong, V. P., and Spyratos, N. (2003). Querying weak instances under extension chase semantics. *International Journal of Comp. Mathematics*, **80(5)**, 591–613.

Levene, M. and Loizou, G. (2003). Why is the snowflake schema a good data warehouse design ? *Information Systems*, **28(3)**, 225–240.

Meo, R., Psaila, G., and Ceri, S. (1997). An extension to sql for mining association rules. In *Data Mining and Knowledge Discovery*, volume 9, pages 275–300.

Turmeaux, T., Cassard, D., Salleb, A., and Vrain, C. (2003). Apprentissage de règles caractéristiques. In *Journées francophones d'Extraction et de gestion des Connaissances (EGC)*, pages 437–448.

Ullman, J. (1988). *Principles of Databases and Knowledge-Base Systems*, volume 1. Computer Science Press.

Summary

The problem of mining *all* frequent queries in a database is intractable, even if we consider conjunctive queries only. In this paper, we show that this problem becomes tractable under the following restrictions : the database scheme is a star scheme ; the data in the database satisfies a set of functional dependencies and a set of referential constraints. We note that star schemes are considered to be the most appropriate for data warehouses, while functional dependencies and referential constraints are the most common constraints that one encounters in real databases. Our approach is based on the weak instance semantics and considers selection-projection queries over weak instances. In such a context, we show that frequent queries can be mined using levelwise algorithms such as Apriori.