Computing Supports of Conjunctive Queries on Relational Tables with Functional Dependencies

Tao-Yuan Jen, Dominique Laurent*
Université Cergy-Pontoise - ETIS - CNRS - F-95000 Cergy-Pontoise
tao-yuan.jen@u-cergy.fr; dominique.laurent@u-cergy.fr

Nicolas Spyratos
Université Paris-Sud - LRI - CNRS - F-91405 Orsay
nicolas.spyratos@lri.fr

Abstract. The problem of mining all frequent queries on a relational table is a problem known to be intractable even for conjunctive queries. In this article, we restrict our attention to conjunctive projection-selection queries and we assume that the table to be mined satisfies a set of functional dependencies. Under these assumptions, we define and characterize two pre-orderings with respect to which the support measure is shown to be anti-monotonic. Each of these pre-orderings induces an equivalence relation for which all queries of the same equivalence class have the same support. The goal of this article is not to provide algorithms for the computation of frequent queries, but rather to provide basic properties of pre-orderings and their associated equivalence relations showing that functional dependencies can be used for an optimized computation of supports of conjunctive queries. In particular, we show that one of the two pre-orderings characterizes anti-monotonicity of the support, while the other one refines the former, but allows to characterize anti-monotonicity with respect to a given table, only. Basic computational implications of these properties are discussed in the article.

Keywords: Functional dependencies, data mining, level-wise algorithms, frequent queries, equivalent queries.
1. Introduction

In this article we address the issue of computing the supports of conjunctive queries on a (relational) table $\Delta$ over an attribute set $U$, where the support of a given query is defined to be the cardinality of its answer in $\Delta$.

This issue is closely related to the problem of mining frequent queries, given that a query is said to be frequent if its support is above a given threshold. This important but challenging problem of data mining, is known to be intractable even for conjunctive queries [8]. Indeed, given a relational table $\Delta$, the size of the search space can be shown to be exponential not only in the number of attributes, but also in the number of tuples (since there are as many selection conditions as there are sub-tuples of tuples in $\Delta$). This problem has attracted significant attention in the last few years [5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17], because of the importance of its potential applications.

As argued in [11], mining all frequent conjunctive queries from a given database is an important issue. Indeed, using the example of [11], consider the well known Internet Movie Database (http://imdb.com) containing almost all possible information about movies, actors and everything related to that, and consider the following queries: first, we ask for all actors that have starred in a movie of the genre ‘drama’; then, we ask for all actors that have starred in a movie of the genre ‘drama’, but that also starred in a (possibly different) movie of the genre ‘comedy’. Now suppose the answer to the first query consists of 1000 actors, and the answer to the second query consists of 900 actors. Obviously, each of these answers considered in isolation, does not necessarily reveal any significant insight, but when combined, these two answers reveal the potentially interesting pattern that actors starring in ‘drama’ movies typically (with a probability of 90%) also star in a ‘comedy’ movie.

In the present work, we consider a fixed attribute set, denoted by $U$, and a fixed set of functional dependencies $FD$ over $U$. In this setting, we denote by $\text{inst}_{FD}(U)$ the set of all relational tables over $U$ that satisfy the functional dependencies in $FD$ and we restrict our attention to conjunctive projection-selection queries. Given a relational table $\Delta$ in $\text{inst}_{FD}(U)$, a conjunctive projection-selection query is of the form $\pi_X(\sigma_{Y=y}(\Delta))$ where $X$ and $Y$ are subsets of $U$ and $y$ is a tuple over $Y$.

We define the support of such a query to be the number of tuples in the answer, i.e., the cardinality of the answer set.

Under these assumptions, we use the set of functional dependencies to define two pre-orderings over queries, and we show that

1. the support measure is anti-monotonic with respect to these two pre-orderings,

2. in the equivalence relations induced by each of these pre-orderings, all queries of the same equivalence class have the same support.

The implications of these results are very important from a computational point of view because of the following:

1. The fact that the support measure is anti-monotonic with respect to the two pre-orderings allows to design algorithms inspired from the well known Apriori algorithm ([1]) for the computation of frequent queries.

2. The fact that all queries in the same equivalence class have the same support implies that only one computation per equivalence class is necessary in order to compute the support of all queries in the class.
Moreover, as one of the major contributions of this article, we show that one of these pre-orderings characterizes anti-monotonicity of the support, independently from the table to be considered, while the other one allows for such a characterization with respect to a given table of \( \text{inst}_{FD}(U) \).

Although this article does not provide algorithms for the computation of frequent projection-selection queries, the work presented in this article can be seen as a generalization of our previous work [13], in which we presented algorithms concerning one of the two pre-orderings of this article, when the table to be mined is the join of all tables in a data warehouse operating from a star schema. We stress that, in this particular case, the complexity of our algorithm in terms of the number of scans of the table is linear in \(|U|\) (i.e., linear in the number of attributes).

Let us illustrate the basic concepts of our approach through an example (that will serve as a running example throughout the article).

**Example 1.1.** Consider the table \( \Delta \) defined over the attribute set \( U = \{ \text{Cid, Cname, Caddr, Pid, Ptype, Qty} \} \), as shown in Figure 1, where:

- \( \text{Cid, Cname and Caddr} \) stand for Customer Identifier, Customer Name and Customer Address,
- \( \text{Pid and Ptype} \) stand for Product Identifier and Product Type,
- \( \text{Qty} \) stands for Quantity (i.e., number of products sold).

Moreover, assume that the table \( \Delta \) satisfies the following set \( FD \) of functional dependencies:

\[
FD = \{ \text{Cid} \rightarrow \text{Cname Caddr}, \text{Pid} \rightarrow \text{Ptype}, \text{Cid Pid} \rightarrow \text{Qty} \}.
\]

One can easily verify that the table \( \Delta \) shown in Figure 1 does satisfy the above dependencies. We note that, in a star schema, the table \( \Delta \) could be the result of joining the following three tables: \( \text{Customer(Cid, Cname, Caddr)}, \text{Product(Pid, Ptype)}, \text{Sales(Cid, Pid, Qty)} \) where \( \text{Customer} \) and \( \text{Product} \) are dimension tables and \( \text{Sales} \) is the fact table.

Assuming that all frequent projection-selection queries are computed using the table \( \Delta \), then we can consider the confidence of rules of the form \( q_1 \Rightarrow q_2 \) (i.e., the ratio of the support of \( q_2 \) over that of \( q_1 \)), where \( q_1 \) and \( q_2 \) are frequent queries such that the support of \( q_2 \) is less than the support of \( q_1 \). To illustrate this point, assume that the queries \( q_1 = \pi_{\text{Cid}}(\sigma_{\text{Caddr}=\text{Paris}}(\Delta)) \) and \( q_2 = \pi_{\text{Cid}}(\sigma_{\text{Caddr Ptype}=\text{Paris beer}}(\Delta)) \) are frequent queries. Then, the fact that the confidence of \( q_1 \Rightarrow q_2 \) is 80% means that 80% of the customers from Paris buy beer.
We give later in the article (see end of Section 3) examples of more general association rules that allow to mine functional dependencies, as well as conditional functional dependencies ([4]).

In this article, we focus only on frequent projection-selection queries, and we leave the study of rules for future work.

Given a query $q$, we denote by $|q|$ the cardinality of the answer set of $q$ in $\Delta$. Then, referring to Figure 1, it is easy to verify that we have $|\pi_{Cid}(\Delta)| \geq |\pi_{Caddr}(\Delta)|$, and that this inequality holds because the table $\Delta$ satisfies the dependency $Cid \rightarrow Caddr$ (which is a consequence of $FD$). Notice that this inequality holds in any table that satisfies the dependencies in $FD$.

Similarly, it is easy to verify that $|\pi_{Cid}(\Delta)| = |\pi_{Cid\backslash Caddr}(\Delta)|$, and that this equality holds because the table $\Delta$ satisfies the dependencies $Cid \rightarrow Cid\backslash Caddr$ and $Cid\backslash Caddr \rightarrow Cid$ (which are consequences of $FD$). Notice again that this equality holds in any table that satisfies the dependencies in $FD$.

We point out that, although not equivalent according to standard query equivalence [18], the queries $\pi_{Cid}(\Delta)$ and $\pi_{Cid\backslash Caddr}(\Delta)$ have the same support. Thus, as long as we are interested in computing supports, only one computation is necessary for these two queries.

One can also easily verify that $|\sigma_{Pid=Ptype=p_2\ beer}(\Delta)| \leq |\sigma_{Pid=p_2}(\Delta)|$, and that this inequality holds in any table because, as $p_2$ is a subtuple of $p_2\ beer$, this is a case of query containment [18].

On the other hand, it is easy to verify that $|\sigma_{Pid=p_2}(\Delta)| \leq |\sigma_{Ptype=\beer}(\Delta)|$, and that this inequality holds because the table $\Delta$ satisfies the dependency $Pid \rightarrow Ptype$ and $p_2$ is associated with $\beer$. Clearly, this inequality holds in any table that satisfies the dependency $Pid \rightarrow Ptype$ and in which the $Pid$ value $p_2$ is associated with the $Ptype$ value $\beer$.

Notice however that this last inequality does not hold in every table that satisfies the dependencies in $FD$. Indeed, consider the table $\Delta'$ obtained from $\Delta$ by replacing all occurrences of $\beer$ by $\wine$. Then $\Delta'$ still satisfies $FD$ but now, $p_2$ is associated with $\wine$ instead of $\beer$. As a result, we have $|\sigma_{Pid=p_2}(\Delta')| > |\sigma_{Ptype=\beer}(\Delta')|$, since the latter answer set in $\Delta'$ is empty.

As shown in the previous example, mining frequent queries can be used for discovering useful knowledge from a given table. We refer to [11] for other examples showing the pertinence of mining frequent queries and corresponding association rules. We emphasize in this respect that such association rules allow to discover unknown functional dependencies and conditional functional dependencies ([4]).

In what follows, based on the observations of Example 1.1, we define two pre-orderings over the set of all projection-selection queries, and show that the support is anti-monotonic with respect to these pre-orderings. Moreover, given an attribute set $U$ and a set of functional dependencies $FD$ over $U$, the following results are shown:

1. The first pre-ordering we consider, denoted by $\preceq$, characterizes the anti-monotonicity of the support, in the sense that for all projection-selection queries $q$ and $q'$, $q \preceq q'$ holds if and only if $|q'| \leq |q|$ holds for all tables $\Delta$ satisfying the given set of functional dependencies $FD$.

2. Given a table $\Delta$ satisfying $FD$, the second pre-ordering we consider, denoted by $\preceq_\Delta$, generalizes the previous one (in the sense that it allows for more comparisons) and characterizes the anti-monotonicity of the support with respect to all tables $\Delta'$ in which the queries under comparison have the same ‘status’. In other words, it is shown that, for all projection-selection queries $q$ and $q'$, $q \preceq_\Delta q'$ holds if and only if $|q'| \leq |q|$ holds for all tables $\Delta'$ satisfying $FD$ and such that $q$ and $q'$ have the same ‘status’ in $\Delta'$ as in $\Delta$ (see Section 4). Roughly, the queries $q$ and $q'$ are said to have the same ‘status’ in $\Delta'$ as in $\Delta$, when the tuples involved in their selection conditions can be
found in $\Delta'$ if and only if they also can be found in $\Delta$ (we refer to Section 4 for a formal definition of the notion of ‘status’).

On the other hand, each of the two pre-orderings induces an equivalence relation over projection-selection queries, and these equivalence relations play a fundamental role in our approach, since two equivalent queries are shown to have the same support. The importance of this property lies in the fact that only one computation is necessary in order to obtain the support of all queries in the same equivalence class.

The article is organized as follows: In Section 2, we briefly review previous work in the area, and in Section 3 we recall basic properties of projection-selection queries. In Section 4, we introduce two pre-orderings for query comparison. In particular, we characterize these two pre-orderings and show that the support measure is anti-monotonic with respect to both of them. In Section 5, we consider the equivalence relations induced by the pre-orderings, we characterize and compare the content of equivalence classes modulo each of the equivalence relations. Section 6 concludes the article and discusses future work.

2. Related Work

The paper [8] considers conjunctive queries, as we do in this article, and points out that without restrictions on the database schema, the problem is intractable. Although some hints on possible restrictions are mentioned in [8], no specific case is studied.

In [9], the authors consider restrictions on the frequent queries to be mined using the formalism of rule based languages. Although the considered class of queries is larger than in our approach, it should be pointed out that, in [9], (i) equivalent queries can be generated that can not be tested efficiently (a problem that does not exist in our approach), and (ii) functional dependencies are not taken into account, as we do in this article.

Comparing now our present contribution to the work in [10, 11] that deals with frequent query computation, we notice that in our approach, we provide two pre-orderings for query comparison that take functional dependencies into account, which is not the case in [10, 11]. As a consequence, we show in this article that these two pre-orderings generalize standard query containment ([18]) used in [10], as well as diagonal containment used in [11].

On the other hand, the work of [10], although dealing with mining tree queries in a graph, is nevertheless closely related to ours. Indeed, in [10], a graph is represented by a binary relation, and frequent tree queries, expressed as SQL queries involving projections, selections and joins, are mined. This work is somehow generalized in [11] to the case of projection-selection-join queries, in which a given relation in the database occurs at most once.

Therefore, the approach in [10, 11] can be considered as being more general than ours because (i) particular joins are taken into account, which is not the case in our approach, and because (ii) for a given join, all frequent projection-selection queries are mined, as we do in our approach. We also note that the consideration of joins, as done in [10, 11], implies that selection conditions involve equalities between two attributes, which we do not consider in this article.

In our previous work [15], we consider the particular case of a database in which relations are organized according to a star schema. In this setting, frequent selection-projection queries are mined from
the associated weak instance ([18]). However, in [15], equivalence classes are defined based on projection only, and thus, only projection queries can be mined through one run of the proposed level-wise algorithm. In our subsequent work [13], we extended these results to projection-selection queries over an arbitrary table with functional dependencies, and we proposed algorithms for mining frequent queries in the case where the considered table is the join of all relations organized according to a star schema. Then, this work has been further extended in [14] in the sense that, in a given star schema, all joins of dimension tables with the fact table are explicitly used in query comparison, based on the key and foreign key constraints.

In [5], a set of attributes, called the key, provides the set of values according to which the supports are to be counted. Then, using a bias language, the different tables involving the key attributes are mined, based on a level-wise algorithm. Our approach (as well as those of [9, 10, 11]) can be seen as a generalization of the work in [5], in the sense that we mine all frequent queries for all keys.

The work of [7] follows roughly the same strategy as in [5], except that joins are first performed in a level-wise manner; and for each join, frequent queries are mined, based also on a level-wise algorithm.

All other approaches dealing with mining frequent queries [6, 12, 16, 17] consider a fixed set of “objects” to be counted during the mining phase and only one table for a given mining task. For instance, in [17], objects are characterized by values over given attributes, whereas in [6], objects are characterized by a query, called the reference. On the other hand, except for [17], all these approaches are restricted to conjunctive queries, as is the case in the present article.

To end this section, we emphasize that, to the best of our knowledge, this work along with that of [13, 14, 15], is the first attempt to consider explicitly constraints on the data set (e.g., functional dependencies or inclusion dependencies) for optimizing the computation of frequent queries.

3. Frequent Queries

3.1. Preliminaries

We assume that the reader is familiar with the relational model for which we follow the notation of [18]. In particular, we consider a fixed attribute set $U$, each attribute $A$ being associated with a domain of values, denoted by $\text{dom}(A)$. For every nonempty subset or schema $X$ of $U$, the domain of $X$ is defined as usual, that is, $\text{dom}(X) = \Pi_{A \in X}(\text{dom}(A))$.

It is important to note that, in this work, every attribute domain is assumed to contain more than one value. In particular, this hypothesis, which is not really a restriction in practice, is required in the proofs of Theorem 4.1 and Theorem 4.2, where tables containing at least two distinct tuples are built up. (Notice however that domains containing a single value are considered in Example 4.2 and in Example 4.4, for the sake of simplification.)

Given two schemas $X$ and $Y$, the schema $X \cup Y$ will be denoted as $XY$ and, similarly, if $A$ is an attribute in $U$, $X \cup \{A\}$ will be denoted by $XA$.

As an additional notational convenience, tuples will be denoted by lowercase characters and their schema by the corresponding uppercase characters. Given a tuple $x$ over $X$, for every subset $Y$ of $X$, $x \cdot Y$ denotes the restriction of $x$ over $Y$.

In this work, we consider a fixed set of functional dependencies over $U$, denoted by $FD$, and we denote by $\text{inst}_{FD}(U)$ the set of all relational tables defined over the attribute set $U$ and satisfying all
dependencies of $FD$. As in [18], we denote by $FD^+$ the set of all functional dependencies that can be inferred from $FD$, based on Armstrong’s axioms ([2]). We recall that these axioms state the following: for all attribute sets $X$, $Y$ and $Z$,

1. **Pseudo-reflexivity**: if $Y \subseteq X$ then infer $X \rightarrow Y$;

2. **Augmentation**: if $X \rightarrow Y$ holds then infer $XZ \rightarrow YZ$;

3. **Transitivity**: if $X \rightarrow Y$ and $Y \rightarrow Z$ hold then infer $X \rightarrow Z$.

Given a table $\Delta$ in $\text{inst}_{FD}(U)$, for every $X \rightarrow Y$ in $FD^+$ and every tuple $x$ over $X$ occurring in $\Delta$, we denote by $\Delta_Y(x)$ the tuple over $Y$ associated in $\Delta$ with $x$ through $X \rightarrow Y$. In other words, $\Delta_Y(x)$ is the (common) $Y$-value of any tuple $t$ in $\Delta$ such that $t.X = x$.

It should be noticed that, if $Y \subseteq X$, then every table $\Delta$ over $U$ satisfies $X \rightarrow Y$ and, for every $x$-value occurring in $\Delta$, $\Delta_Y(x)$ is the restriction of $x$ over $Y$, that is, $\Delta_Y(x) = x.Y$.

Referring back to Figure 1, $\text{Cid} \rightarrow \text{Caddr}$ is in $FD^+$, and we have, for instance, $\Delta_{\text{Caddr}}(c_1) = \text{Paris}$, as all tuples $t$ such that $t.\text{Cid} = c_1$ satisfy $t.\text{Caddr} = \text{Paris}$.

In this article, given an attribute set $X$, we consider the notion of closure of $X$ as in [18], that is: $X^+$ denotes the closure of $X$ (with respect to $FD$), namely, the set of all attributes $A$ in $U$ such that the dependency $X \rightarrow A$ is in $FD^+$. A schema $X$ such that $X = X^+$ is said to be closed.

The following lemma, that will be used for the characterization of what we call equivalent queries (see Section 5), relates schemas having the same closure to the functions induced by the functional dependencies of $FD$.

**Lemma 3.1.** Let $\Delta$ be in $\text{inst}_{FD}(U)$, and $X_1$ and $X_2$ two schemas such that $X_1^+ = X_2^+$. For all tuples $x_1$ and $x_2$ over $X_1$ and $X_2$, respectively, we have $x_2 = \Delta_{X_2}(x_1)$ if and only if $x_1 = \Delta_{X_1}(x_2)$.

**Proof:**
Due to the symmetry of the statement in the lemma, we only prove the implication $x_2 = \Delta_{X_2}(x_1) \Rightarrow x_1 = \Delta_{X_1}(x_2)$. Let $t$ in $\Delta$ such that $t.X_2 = x_2$, then, as $X_2 \rightarrow X_1 \in FD^+$ (because $X_1^+ = X_2^+$), $t.X_1 = \Delta_{X_1}(x_2)$. Similarly, let $t'$ in $\Delta$ such that $t'.X_1 = x_1$, then, as $X_1 \rightarrow X_2 \in FD^+$ (because $X_1^+ = X_2^+$), $t'.X_2 = \Delta_{X_2}(x_1)$. Thus, as we assume that $x_2 = \Delta_{X_2}(x_1)$, we have $t'.X_2 = x_2$. Therefore, $t'.X_2 = t.X_2$, which implies that $t.X_1 = t'.X_1$ (since $X_2 \rightarrow X_1 \in FD^+$). Consequently, $x_1 = \Delta_{X_1}(x_2)$, which completes the proof. 

Moreover, functional dependencies allow for comparisons of the cardinalities of the answers to queries, as stated in the following lemma.

**Lemma 3.2.** For all nonempty schemas $X$ and $Y$ such that $X \rightarrow Y$ is in $FD^+$ and for every table $\Delta$ in $\text{inst}_{FD}(U)$, we have:

1. $|\pi_X(\Delta)| \geq |\pi_Y(\Delta)|$ and
2. $|\sigma_{X = x}(\Delta)| \leq |\sigma_{Y = \Delta_{Y}(x)}(\Delta)|$. 
Proof:
1. Since $\Delta$ satisfies $X \rightarrow Y$, there exists a total, onto function from $\pi_X(\Delta)$ to $\pi_Y(\Delta)$. Therefore, $|\pi_X(\Delta)| \geq |\pi_Y(\Delta)|$.

2. For every $t$ in $\sigma_{X=x}(\Delta)$, $t.X = x$. Thus, we have $t.Y = \Delta_Y(x)$, showing that $t$ is in $\sigma_{Y=\Delta_Y(x)}(\Delta)$. Therefore, we have $|\sigma_{X=x}(\Delta)| \leq |\sigma_{Y=\Delta_Y(x)}(\Delta)|$, and the proof is complete. $\square$

In the standard relational model, schemas are assumed to be nonempty sets. However, for the purposes of this article, we also consider the empty schema, denoted by $\emptyset$. This is so because, as will be seen later on:

1. Queries reduced to a projection only will be considered as projection-selection queries with a selection condition holding over the empty attribute set.

2. Projections over the empty attribute set make sense in our approach, in particular when considering equivalent queries.

Contrary to the case of nonempty attribute sets, the set $\text{dom}(\emptyset)$ is assumed to contain a single tuple, namely the empty tuple, denoted by $\top$. We consider $\top$ as being a subtuple of any tuple, and, given a table $\Delta$ over $U$, we have:

- $\sigma_{\emptyset = \top}(\Delta) = \Delta$, meaning that the selection condition $\emptyset = \top$ always evaluates to true.
- If $\Delta = \emptyset$ then $\pi_{\emptyset}(\Delta) = \emptyset$, as any projection of an empty table results in an empty table. Otherwise $\pi_{\emptyset}(\Delta) = \{\top\}$, as in this case the projection should not be empty and the domain of the empty schema is reduced to the single tuple $\top$.

Moreover, we consider functional dependencies involving $\emptyset$ along with the following rules, which can be shown to be consistent with the Armstrong axioms.

- For every $X$ (possibly empty), every table $\Delta$ satisfies $X \rightarrow \emptyset$. Thus, for every $x$ in $\text{dom}(X)$, $\top = \Delta_X(x)$.

- For every set of functional dependencies $FD$, $\emptyset^+ = \emptyset$.

We note however that, in this article, we do not consider functional dependencies of the form $\emptyset \rightarrow X$, which are satisfied by a table $\Delta$ if $\pi_X(\Delta)$ is reduced to a single tuple. This is so because, in practice, these dependencies are usually not considered.

On the other hand, some of the proofs in this article are based on the characterization of functional dependency satisfaction given in Proposition 3.1 below, which is an immediate consequence of Theorem 6.1 of [3]. In order to state this proposition, we introduce the following notation: for all tuples $x_1$ and $x_2$ over $X_1$ and $X_2$, respectively, $\text{match}(x_1, x_2)$ denotes the set of all attributes $A$ of $X_1 \cap X_2$ such that $x_1.A = x_2.A$.

\textbf{Proposition 3.1.} Given an attribute set $U$ over which the set $FD$ of functional dependencies is assumed, let $\Delta$ be a table over $U$. Then, $\Delta$ is in $\text{inst}_{FD}(U)$ if and only if for all $t$ and $t'$ in $\Delta$, $\text{match}(t, t')$ is closed, i.e., $\text{match}(t, t') = (\text{match}(t, t'))^+$. 

3.2. Queries

The projection-selection queries considered in our approach are standard relational conjunctive projection-selection queries.

Definition 3.1. A conjunctive selection condition, or a selection condition for short, is an equality of the form \( Y = y \) where \( Y \) is a possibly empty relation schema and \( y \) a tuple in \( \text{dom}(Y) \). Let \( S = (Y = y) \) be a selection condition. A tuple \( t \) over \( U \) is said to satisfy \( S \), denoted by \( t \models S \), if: either \( Y = \emptyset \) and \( y = T \), or \( Y \neq \emptyset \) and \( t.Y = y \).

We denote by \( Q(U) \) the set of all queries of the form \( \pi_X(\sigma_{Y=y}(\delta)) \) where \( \delta \) stands for an arbitrary table over \( U \), \( X \) and \( Y \) are possibly empty subsets of \( U \), and \( y \) is a tuple in \( \text{dom}(Y) \). For the sake of simplicity, every query \( \pi_X(\sigma_{Y=y}(\delta)) \) in \( Q(U) \) is denoted by \( \pi_X\sigma_y(\delta) \), where the schema \( Y \) of \( y \) is understood.

Given a table \( \Delta \) in \( inst_{FD}(U) \) and a query \( q = \pi_X\sigma_y(\delta) \) in \( Q(U) \), the answer to \( q \) in \( \Delta \), denoted by \( ans_\Delta(q) \), is defined as usual, i.e., as being the set of those tuples \( x \) over \( X \) such that there exists a tuple \( t \) in \( \Delta \) such that \( x = t.X \) and \( t \models (Y = y) \).

Moreover, \( Q(\Delta) \) denotes the set of all queries \( \pi_X\sigma_y(\delta) \) of \( Q(U) \) such that \( y \in \pi_Y(\Delta) \).

We draw attention on the fact that, in the remainder of the article, according to the notational convention mentioned at the beginning of the present section, for every query of the form \( \pi_X\sigma_y(\delta) \), \( Y \) will denote the schema of the tuple \( y \).

We note from Definition 3.1 that all queries in \( Q(U) \) of the form \( \pi_X\sigma_y(\delta) \), where \( X \) is a nonempty schema, are simply the relational queries of the form \( \pi_X(\delta) \). Consequently \( \pi_U\sigma_T(\delta) \) stands for the simple query \( \delta \).

Moreover, for every \( \Delta \) in \( inst_{FD}(U) \) and every \( q \) in \( Q(U) \setminus Q(\Delta) \), \( ans_\Delta(q) = \emptyset \). In particular, if \( y \in \pi_Y(\Delta) \) then \( ans_\Delta(\pi_0\sigma_y(\delta)) = \{T\} \), else \( ans_\Delta(\pi_0\sigma_y(\delta)) = \emptyset \).

Example 3.1. Referring back to Example 1.1, \( \pi_{\text{Cid}}\sigma_{Y}(\delta) \) and \( \pi_{\text{Addr}}\sigma_{\text{Paris beer}}(\delta) \) are queries in \( Q(\Delta) \), and we have: \( ans_\Delta(\pi_{\text{Cid}}\sigma_{Y}(\delta)) = \{c_1, c_2, c_3, c_4\} \), and \( ans_\Delta(\pi_{\text{Addr}}\sigma_{\text{Paris beer}}(\delta)) = \{\text{Paris}\} \). Similarly, \( \pi_{\text{Quad}}\sigma_{\text{Paris beer}}(\delta) \) is also a query in \( Q(\Delta) \), and we have \( ans_\Delta(\pi_{\text{Quad}}\sigma_{\text{Paris beer}}(\delta)) = \{T\} \).

On the other hand, the queries \( \pi_{\text{Cid}}\sigma_{\text{NY milk}}(\delta) \) and \( \pi_{\text{NY milk}}(\delta) \) are in \( Q(U) \) but not in \( Q(\Delta) \), because \( \text{NY milk} \) is not in \( \pi_{\text{Addr}_{\text{Type}}}(\Delta) \). Thus, \( ans_\Delta(\pi_{\text{Cid}}\sigma_{\text{NY milk}}(\delta)) = ans_\Delta(\pi_{\text{NY milk}}(\delta)) = \emptyset \).

The notion of frequent query in our approach is defined in much the same way as in [8, 11].

Definition 3.2. Let \( \Delta \) be in \( inst_{FD}(U) \) and \( q \) be in \( Q(U) \). The support of \( q \) in \( \Delta \), denoted by \( sup_\Delta(q) \), is the cardinality of the answer to \( q \) in \( \Delta \), i.e., \( sup_\Delta(q) = |ans_\Delta(q)| \).

Given a support threshold \( \text{min-sup} \), a query \( q \) is said to be frequent in \( \Delta \) if \( sup_\Delta(q) \geq \text{min-sup} \).

We notice that for every \( \Delta \) in \( inst_{FD}(U) \) and every query \( q \) in \( Q(U) \), we have \( 0 \leq sup_\Delta(q) \leq |\Delta| \). Moreover, for every \( q \) in \( Q(U) \), we have that \( q \not\in Q(\Delta) \) if and only if \( sup_\Delta(q) = 0 \).

Indeed, if \( q = \pi_X\sigma_y(\delta) \) is in \( Q(U) \setminus Q(\Delta) \), then \( \Delta \) contains no tuple \( t \) such that \( t.Y = y \), which shows that \( sup_\Delta(q) = 0 \). Conversely, if \( q \in Q(\Delta) \), then \( \Delta \) contains at least one tuple \( t \) such that \( t.Y = y \), and thus, \( sup_\Delta(q) \geq 1 \).

As a consequence, when it comes to compute all frequent queries, those in \( Q(U) \setminus Q(\Delta) \) are not relevant since their support is 0, a value meant to be less than the support threshold \( \text{min-sup} \).
On the other hand, if \( q = \pi_X \sigma_y(\delta) \) is in \( Q(\Delta) \) and \( Y \rightarrow X \) is in \( FD^+ \), then we have \( sup_\Delta(q) = 1 \). This is so because, if \( sup_\Delta(q) > 1 \) then \( \Delta \) contains at least two tuples \( t \) and \( t' \) such that \( t.Y = t'.Y = y \) and \( t.X \neq t'.X \), which is a violation of the dependency \( Y \rightarrow X \) assumed to be in \( FD^+ \).

Referring back to Example 1.1, it is easy to see that \( sup_\Delta(q_1) = 3 \) and \( sup_\Delta(q_2) = 2 \). Thus, for a support threshold equal to 3, \( q_1 \) is frequent whereas \( q_2 \) is not.

Before ending this section, we point out that having computed all frequent queries of \( Q(U) \), it is possible to mine functional dependencies other than those in \( FD^+ \), as well as conditional functional dependencies ([4]).

Indeed, given a table \( \Delta \), it can be seen that if in \( \Delta \), the confidence of the rule \( \pi_{X_1}, X_2(\delta) \Rightarrow \pi_{X_1}(\delta) \) is 1 (i.e., if the ratio of \( \text{sup}_\Delta(\pi_{X_1}(\delta)) \) over \( \text{sup}(\pi_{X_1}, X_2(\delta)) \) is equal to 1), then \( \Delta \) satisfies the functional dependency \( X_1 \rightarrow X_2 \). This is so because, if the confidence of the rule \( \pi_{X_1}, X_2(\delta) \Rightarrow \pi_{X_1}(\delta) \) is 1, then \( \text{sup}_\Delta(\pi_{X_1}, X_2(\delta)) = \text{sup}_\Delta(\pi_{X_1}(\delta)) \), meaning that the answers in \( \Delta \) of \( \pi_{X_1}, X_2(\delta) \) and \( \pi_{X_1}(\delta) \) have the same cardinality.

Therefore, this implies that, in the table \( \pi_{X_1}, X_2(\Delta) \), there exists a function from the set of \( X_1 \)-values to the set of \( X_2 \)-values. In other words, this means that \( \Delta \) satisfies the functional dependency \( X_1 \rightarrow X_2 \).

Regarding conditional functional dependencies, we recall from [4] that such a dependency is denoted by \( (\Delta : X_1 \rightarrow X_2, T) \), where \( \Delta \) is a table, \( X_1 \) and \( X_2 \) are subsets of the schema \( U \) of \( \Delta \), and \( T \) is a specific table over \( U \) containing partial tuples (i.e., tuples with constants and null values) that specify selection conditions.

For every \( \tau \) in \( T \), let \( Y_1^\tau = y_1^\tau \) and \( Y_2^\tau = y_2^\tau \) be the two selection conditions induced by \( \tau \) on \( X_1 \) and \( X_2 \), respectively (i.e., for \( i = 1, 2 \), \( y_i^\tau \) is the largest total sub-tuple of \( \tau \) such that \( Y_i^\tau \subseteq X_i \)). Then, \( \Delta \) is said to satisfy \( (\Delta : X_1 \rightarrow X_2, T) \) if for every \( \tau \) in \( T \) and for all tuples \( t \) and \( t' \) in \( \Delta \) such that \( t.X_1 = t'.X_1 \) and \( t.Y_1^\tau = t'.Y_1^\tau = y_1^\tau \), then \( t.X_2 = t'.X_2 \) and \( t.Y_2^\tau = t'.Y_2^\tau = y_2^\tau \).

Consequently, it can be seen that \( \Delta \) satisfies \( (\Delta : X_1 \rightarrow X_2, T) \) if and only if, for every \( \tau \) in \( T \), (i) the table \( \sigma y_1^\tau(\Delta) \) satisfies \( X_1 \rightarrow X_2 \) and (ii) \( \sigma y_1^\tau(\Delta) \subseteq \sigma y_2^\tau(\Delta) \).

Therefore, \( \Delta \) satisfies \( (\Delta : X_1 \rightarrow X_2, T) \) if and only if, for every \( \tau \) in \( T \), the confidences of the rules (i) \( \pi_{X_1}, X_2(\sigma y_1^\tau(\delta)) \Rightarrow \pi_{X_1}(\sigma y_1^\tau(\delta)) \) and (ii) \( \sigma y_1^\tau(\delta) \Rightarrow \sigma y_2^\tau(\delta) \) are both equal to 1.

4. Query Comparison

In this section, we define and characterize two pre-orderings so as to compare queries of \( Q(U) \). The first one is shown to characterize the anti-monotonicity of the support when considering all tables in \( \text{inst}_{FD}(U) \). Then, we refine this first pre-ordering by considering a fixed table of \( \text{inst}_{FD}(U) \).

4.1. Query Comparison Characterizing Anti-monotonicity

**Definition 4.1.** Let \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X, y_1(\delta) \) in \( Q(U) \), \( q_1 \) is said to be more specific than \( q \), denoted by \( q \preceq q_1 \), if

1. \( XY_1 \rightarrow X_1 \) is in \( FD^+ \), and
2. \( y \) is a subtuple of \( y_1 \).

It is easy to see from Definition 4.1 that for all possibly empty schemas \( X \) and \( X_1 \) and every tuple \( y \), we have:
• If $y$ is a subtuple of $y_1$, $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_{y_1}(\delta)$ (because $XY_1 \rightarrow X$ is a trivial dependency). In particular, $\pi_X \sigma_T(\delta) \preceq \pi_X \sigma_{y_1}(\delta)$ (because $\top$ is a subtuple of any tuple $y_1$).

• If $X \rightarrow X_1$ is in $FD^+$ then $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_{y_1}(\delta)$ (because $XY \rightarrow X_1$ can be deduced from $X \rightarrow X_1$). In particular, $\pi_X \sigma_T(\delta) \preceq \pi_Y \sigma_{y_1}(\delta)$ (because $X \rightarrow \emptyset$ is assumed to be in any $FD^+$).

• If $Y \rightarrow X$ is in $FD^+$, $\pi_\emptyset \sigma_T(\delta) \preceq \pi_X \sigma_y(\delta)$ (because in this case $\emptyset Y \rightarrow X$ is in $FD^+$).

It is also important to note that, in this case, we also have $\pi_X \sigma_y(\delta) \preceq \pi_\emptyset \sigma_y(\delta)$ (because $XY \rightarrow \emptyset$ is assumed to be in any $FD^+$) and $\pi_\emptyset \sigma_y(\delta) \preceq \pi_X \sigma_y(\delta)$ (because $Y \rightarrow X$ is assumed to be in $FD^+$).

• $\pi_U \sigma_T(\delta) \preceq \pi_X \sigma_y(\delta)$ (because $\top$ is a subtuple of any tuple $y$, and because $UY = U$ and $X \subseteq U$ entail that $UY \rightarrow X$ is in any $FD^+$). This means that $\pi_U \sigma_T(\delta)$ (i.e., $\delta$) is the less specific query in $Q(U)$.

• If $u$ is a tuple over $U$ and $y$ is a subtuple of $u$, $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_u(\delta)$ (because $y$ is a subtuple of $u$, and because $UX = U$ and $X \subseteq U$ entail that $UX \rightarrow X_1$ is in any $FD^+$).

The following proposition shows that the relation $\preceq$ is a pre-ordering over the set $Q(U)$, i.e., that $\preceq$ is reflexive and transitive.

**Proposition 4.1.** The relation $\preceq$ is a pre-ordering over $Q(U)$.

**Proof:**
First, it is easy to see that $\preceq$ is reflexive. So, we only prove the transitivity of $\preceq$. To this end, let $q = \pi_X \sigma_y(\delta)$, $q_1 = \pi_X \sigma_{y_1}(\delta)$ and $q_2 = \pi_X \sigma_{y_2}(\delta)$ be in $Q(U)$ such that $q \preceq q_1$ and $q_1 \preceq q_2$. Then, by Definition 4.1: $XY_1 \rightarrow X_1$ and $X_1Y_2 \rightarrow X_2$ are in $FD^+$, and $y$ is a subtuple of $y_1$ and $y_1$ is a subtuple of $y_2$. Thus, $y$ is a subtuple of $y_2$, and, as $Y_1 \subseteq Y_2$, $XY_2 \rightarrow XY_1$ is in $FD^+$. Therefore, $XY_2 \rightarrow X_1Y_2$ is in $FD^+$ (because $XY_1 \rightarrow X_1 \in FD^+$), and so, $XY_2 \rightarrow X_2$ is in $FD^+$ (because $X_1Y_2 \rightarrow X_2 \in FD^+$). Hence, $q \preceq q_2$, and the proof is complete.

Notice that $\preceq$ is not a partial ordering over $Q(U)$, as is shown in the following example.

**Example 4.1.** Referring back to Example 1.1, $q = \pi_{\text{CidCaddr}}(\delta)$ and $q_1 = \pi_{\text{CaddrCid}}(\delta)$ are distinct queries of $Q(U)$ and we have:

• $q \preceq q_1$ (because $\text{Cid} \rightarrow \text{CidCaddr}$ can be deduced from $FD$),

• $q_1 \preceq q$ (because $\text{CaddrCid} \rightarrow \text{Cid}$ can be trivially deduced from $FD$).

The following example shows an exhaustive graph of query comparisons in a simple case.

**Example 4.2.** Let us consider the simple case where $U$ contains only two attributes $A$ and $B$ and $FD$ is empty. Contrary to our hypothesis on the cardinalities of attribute domains, we assume that $\text{dom}(A) = \{a\}$ and $\text{dom}(B) = \{b\}$, for the sake of simplification. In this case, $Q(U)$ contains all queries $\pi_X \sigma_y(\delta)$ where $X \in \{\emptyset, A, B, AB\}$ and $y \in \{\top, a, b, ab\}$.
The pre-ordering ≤ allows to compare these queries according to the graph shown below, where a top-down link between two queries \( q \) and \( q_1 \) should be read as \( q \preceq q_1 \), and where two queries \( q \) and \( q_1 \) in the same box are such that both \( q \preceq q_1 \) and \( q_1 \preceq q \) hold.

\[ \begin{array}{c}
\pi_{AB}\sigma_T(\delta) \\
\pi_A\sigma_T(\delta) \\
\pi_{AB}\sigma_b(\delta), \pi_A\sigma_b(\delta) \\
\pi_{AB}\sigma_b(\delta) \\
\pi_A\sigma_b(\delta) \\
\pi_A\sigma_{ab}(\delta), \pi_{AB}\sigma_{ab}(\delta), \pi_{AB}\sigma_{ab}(\delta) \\
\pi_A\sigma_{ab}(\delta) \end{array} \]

We note that replacing item 1 of Definition 4.1 by \( X \rightarrow X_1 \in FD^+ \) would also lead to a partial pre-ordering according to which the support is anti-monotonic. However this pre-ordering would be strictly more specific than ≤, because \( XY_1 \rightarrow X_1 \in FD^+ \) does not imply \( X \rightarrow X_1 \in FD^+ \).

For instance, in the context of our running example, \( \pi_{Cid}\sigma_T(\delta) \) and \( \pi_{Qty}\sigma_{p_2}(\delta) \) would not be comparable, whereas \( \pi_{Cid}\sigma_T(\delta) \preceq \pi_{Qty}\sigma_{p_2}(\delta) \) holds because \( CidPid \rightarrow Qty \in FD^+ \).

Moreover, it can be seen that the pre-ordering ≤ generalizes query containment ([18]) and diagonal containment ([11]). Indeed, denoting these pre-orderings by \( \sqsubseteq \) and \( \sqsubseteq^D \), respectively, given two queries \( q = \pi_X\sigma_{y}(\delta) \) and \( q_1 = \pi_{X1}\sigma_{y_1}(\delta) \) in \( Q(U) \), in our context, we have

- \( q_1 \sqsubseteq^D q \) if and only if \( X = X_1 \) and \( y \) is a subtuple of \( y_1 \), which implies that \( q \preceq q_1 \).
- According to [11], \( q_1 \sqsubseteq^D q \) holds if and only if \( q_1 \sqsubseteq \pi_{X1}(q) \), that is, if and only if \( X_1 \subseteq X \) and \( y \) is a subtuple of \( y_1 \). This again implies that \( q \preceq q_1 \).

We note that even in the case where \( FD = \emptyset, \preceq \) is still strictly more general than \( \sqsubseteq^D \), because in this case, for all \( q = \pi_X\sigma_{y}(\delta) \) and \( q_1 = \pi_{X1}\sigma_{y_1}(\delta) \) in \( Q(U) \), we have \( q \preceq q_1 \) if \( X_1 \subseteq XY_1 \) and \( y \) is a subtuple of \( y_1 \), which is implied by but not equivalent to the fact that \( q_1 \sqsubseteq^D q \).

As an example, referring back to Example 1.1, \( \pi_{Cid}\sigma_T(\delta) \preceq \pi_{Cname}\sigma_{John}(\delta) \) holds (because \( Cname \subseteq Cid\ Cname \)), whereas these queries are not comparable according to \( \sqsubseteq^D \).

This implies that, even if functional dependencies are not considered, as done in [11], diagonal containment is not the optimal way of comparing queries, since more comparisons are possible according to our pre-ordering ≤. Moreover, as shown in Theorem 4.1 below, ≤ is optimal in this respect, that is, \( q \preceq q_1 \) is equivalent to the fact that \( sup_{\Delta}(q_1) \leq sup_{\Delta}(q) \) holds in every \( \Delta \) of \( inst_{FD}(U) \).

**Theorem 4.1.** For all queries \( q \) and \( q_1 \) in \( Q(U) \), \( q \preceq q_1 \) holds if and only if, for every \( \Delta \) in \( inst_{FD}(U) \), \( sup_{\Delta}(q_1) \leq sup_{\Delta}(q) \).
Proof:
Let \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X \sigma_{y_1}(\delta) \) be in \( Q(U) \). We first show that, given a table \( \Delta \) in \( \text{inst}_{FD}(U) \), if \( q \preceq q_1 \), then \( \sup_{\Delta}(q_1) \leq \sup_{\Delta}(q) \).

- If \( y_1 = \top \), as we assume that \( y \) is a subtuple of \( y_1 \), we have \( y = \top \). Thus, the result directly follows from Lemma \( 3.2(1) \), because in this case, \( X \rightarrow X_1 \in FD^+ \).

- If \( y_1 \neq \top \), then, the case where \( y_1 \notin \pi_Y(\Delta) \) is trivial because we have \( \sup_{\Delta}(q_1) = 0 \). So, let us assume that \( y_1 \in \pi_Y(\Delta) \). In this case, the table \( \sigma_{y_1}(\Delta) \) is not empty and satisfies \( X \rightarrow Y_1 \). As \( XY_1 \rightarrow X_1 \) is assumed to be in \( FD^+ \), \( \sigma_{y_1}(\Delta) \) satisfies \( X \rightarrow X_1 \). Thus, by Lemma \( 3.2(1) \), we obtain that \( |\pi_X \sigma_{y_1}(\Delta)| \leq |\pi_X \sigma_y(\Delta)| \). On the other hand, as \( y \) is a subtuple of \( y_1 \), \( |\pi_X \sigma_{y_1}(\Delta)| \leq |\pi_X \sigma_y(\Delta)| \), and thus, we obtain \( |\text{ans}(q_1)| \leq |\text{ans}(q)| \). Therefore, it follows that \( \sup_{\Delta}(q_1) \leq \sup_{\Delta}(q) \).

Conversely, we proceed by contraposition, namely, assuming that one of the two items in Definition \( 4.1 \) is not satisfied, we construct a table \( \Delta \) in \( \text{inst}_{FD}(U) \) in which \( \sup_{\Delta}(q_1) > \sup_{\Delta}(q) \).

- Let us first assume that \( y \) is not a subtuple of \( y_1 \). In this case, as \( y \neq \top \) we have \( Y \neq \emptyset \). Thus there exists an attribute \( A \) in \( Y \) such that, either \( A \) does not belong to \( Y_1 \), or \( A \) is in \( Y_1 \) and \( y.A \neq y.A \). Let \( \Delta \) be a table over \( U \) containing a single tuple \( t \) such that \( t.Y_1 = y_1 \) and \( t.A \neq y.A \). Then, clearly, \( \Delta \) is in \( \text{inst}_{FD}(U) \) (because a table containing a single tuple satisfies any set of functional dependencies), and we have \( \sup_{\Delta}(q_1) = 1 \) (because \( \text{ans}(q_1) = \{t.X_1\} \) and \( \sup_{\Delta}(q) = 0 \) (because, as \( t.Y \neq y \), \( \text{ans}(q) = \emptyset \)). Thus, \( \sup_{\Delta}(q_1) > \sup_{\Delta}(q) \).

- Let us now assume that \( y \) is a subtuple of \( y_1 \), but that \( XY_1 \rightarrow X_1 \) is not in \( FD^+ \). In this case, we consider a table \( \Delta \) over \( U \) containing two tuples \( t_1 \) and \( t_2 \) defined as follows:

\[
\begin{align*}
- t_1.(XY_1)^+ &= t_2.(XY_1)^+, \\
- t_1.Y_1 &= t_2.Y_1 = y_1, \\
- \text{for every attribute } A \text{ not in } (XY_1)^+, t_1.A &\neq t_2.A.
\end{align*}
\]

We note that there exist attributes not in \( (XY_1)^+ \) because, assuming that \( (XY_1)^+ = U \) entails \( XY_1 \rightarrow X_1 \in FD^+ \). Then, as \( \text{match}(t_1, t_2) = (XY_1)^+ \), by Proposition \( 3.1 \), \( \Delta \) satisfies \( FD \). Moreover, we have \( \sup_{\Delta}(q) = 1 \) because, \( t_1.X = t_2.X \) and, since \( y \) is a subtuple of \( y_1 \) and \( t_1.Y_1 = t_2.Y_1 = y_1 \), we have \( t_1.Y = t_2.Y = y \).

On the other hand, we have \( t_1.X_1 \neq t_2.X_1 \) because \( X_1 \nsubseteq (XY_1)^+ \). Indeed, if \( X_1 \nsubseteq (XY_1)^+ \), then \( XY_1 \rightarrow X_1 \) is in \( FD^+ \), which is a contradiction to our hypothesis. Since \( t_1.Y_1 = t_2.Y_1 = y_1 \), we obtain that \( \sup_{\Delta}(q_1) = 2 \), and thus that \( \sup_{\Delta}(q_1) > \sup_{\Delta}(q) \). Therefore, the proof is complete. \( \Box \)

We recall that showing that the support is anti-monotonic with respect to a specificity relation is a basic property that allows to design level-wise algorithms (such as Apriori [1]), in order to compute all frequent queries, given a table in \( \text{inst}_{FD}(U) \).

Now, although Theorem \( 4.1 \) provides an important and generic characterization of anti-monotonicity of the support of queries, it fails to capture many basic comparisons when a specific table of \( \text{inst}_{FD}(U) \) is considered, which is the case in practice.

For instance, we recall from Example \( 1.1 \) that, in the given table \( \Delta \), we have \( \sup_{\Delta}(\pi_U \sigma_{p_2}(\delta)) \leq \sup_{\Delta}(\pi_U \sigma_{\text{beer}}(\delta)) \). However, these two queries are not comparable according to \( \preceq \) because none of the tuples \( p_2 \) and \( \text{beer} \) is a subtuple of the other. We address this important issue next.
4.2. Query Comparison Based on a Given Table in $inst_{FD}(U)$

In order to cope with the problem mentioned above, we assume a fixed table $\Delta$ in $inst_{FD}(U)$ and we define the following pre-ordering (which is an extension of the one given in [13] to all queries in $Q(U)$).

**Definition 4.2.** Let $\Delta$ be in $inst_{FD}(U)$ and $q = \pi_X \sigma_y(\delta)$ and $q_1 = \pi_X \sigma_{y_1}(\delta)$ be in $Q(U)$. $q_1$ is said to be more specific than $q$ with respect to $\Delta$, denoted by $q \preceq_\Delta q_1$, if one of the following cases holds:

1. $q_1$ is in $Q(U)$ but not in $Q(\Delta)$,
2. $q$ and $q_1$ are in $Q(\Delta)$ and $Y_1 \rightarrow X_1 \in FD^+$,
3. $q$ and $q_1$ are in $Q(\Delta)$ such that $XY_1 \rightarrow X_1 \in FD^+$, $Y_1 \rightarrow Y \in FD^+$, and $y = \Delta_Y(y_1)$.

It should be noticed from Definition 4.2 that, contrary to $\preceq$, the pre-ordering $\preceq_\Delta$ is instance dependent, in the sense that testing whether $q \preceq_\Delta q_1$ holds requires to access the table $\Delta$. Indeed, given $q = \pi_X \sigma_y(\delta)$ and $q_1 = \pi_X \sigma_{y_1}(\delta)$ in $Q(U)$, in order to know that $q \preceq_\Delta q_1$, one must check that:

1. In case 1, $y_1$ does not belong to $\pi_Y(\Delta)$.
2. In case 2, $y$ and $y_1$ belong to $\pi_Y(\Delta)$ and $\pi_Y(\Delta)$, respectively.
3. In case 3, $y$ and $y_1$ belong to $\pi_Y(\Delta)$ and $\pi_Y(\Delta)$, respectively, and $y = \Delta_Y(y_1)$, which is equivalent to the fact that $yy_1$ belongs to $\pi_Y(\Delta)$.

We point out that these tests are linear in the size of $\Delta$, because a simple scan of $\Delta$ allows to conclude on whether $q \preceq_\Delta q_1$ holds. Moreover, it can be seen from [13] that, in our algorithms, such tests are even not necessary in order to mine all frequent queries, because only queries in $Q(\Delta)$ are considered.

The following example shows various comparisons according to $\preceq_\Delta$.

**Example 4.3.** In the context of Example 1.1, we have:

- $\pi_{Cid} \sigma_{Paris}(Y) \preceq_\Delta \pi_{Cid} \sigma_{Paris} \sigma_{beery}(Y)$, because the two queries are in $Q(\Delta)$, $Cid Caddr Ptype \rightarrow Cid$ and $Caddr Ptype \rightarrow Caddr$ are in $FD^+$, and $\Delta_{Cadd} Y (Paris) = Paris$.

  Notice that, in this case, we also have $\pi_{Cid} \sigma_{Paris}(Y) \preceq \pi_{Cid} \sigma_{Paris beery}(Y)$.

- $\pi_{Cid} \sigma_{\Delta}(Y) \preceq_\Delta \pi_{Cid} \sigma_{Paris}(Y)$, because the two queries are in $Q(\Delta)$, $Cid Caddr \rightarrow Cid$ and $Cadd Y \rightarrow \emptyset$ are in $FD^+$, and $\Delta_0 Y (Paris) = \top$.

  In this case again, we also have $\pi_{Cid} \sigma_{\Delta}(Y) \preceq \pi_{Cid} \sigma_{Paris}(Y)$.

- $\pi_{Cid} \sigma_{beery}(Y) \preceq_\Delta \pi_{Qty} \sigma_{P}\sigma_{p_2} \sigma_{beery}(Y)$, because the two queries are in $Q(\Delta)$, $Cid Pid \rightarrow Qty$ and $Pid \rightarrow Ptype$ are in $FD^+$, and $\Delta_{Ptype}(P_2) = beery$.

  In this case however, $\pi_{Cid} \sigma_{beery}(Y)$ and $\pi_{Qty} \sigma_{P}\sigma_{p_2} \sigma_{beery}(Y)$ are not comparable according to $\preceq_\Delta$.

- $\pi_{Cid} \sigma_{beery}(Y) \preceq_\Delta \pi_{Ptype} \sigma_{P}\sigma_{p_2} \sigma_{beery}(Y)$, because the two queries are in $Q(\Delta)$, and $Cid Pid \rightarrow Ptype$ is in $FD^+$.

  As above, $\pi_{Cid} \sigma_{beery}(Y)$ and $\pi_{Ptype} \sigma_{P}\sigma_{p_2} \sigma_{beery}(Y)$ are not comparable according to $\preceq_\Delta$. 
More generally, given a table \( \Delta \), the two distinct queries
\( \preceq \), as was the case for the pre-ordering \( \preceq \). Assuming now that
\( y \) and \( \gamma \), and thus, by Proposition 4.2, \( q \preceq q_1 \) and \( q_1 \preceq q \).

The following example shows how the pre-ordering \( \preceq \) behaves in the simple case of Example 4.2.
**Example 4.4.** As in Example 4.2, let $U = \{A, B\}$, $FD = \emptyset$, and, contrary to our hypothesis on the cardinalities of attribute domains, assume that $dom(A) = \{a\}$ and $dom(B) = \{b\}$, for the sake of simplification. In this case, considering the database cardinalities of attribute domains, assume that $sup_\Delta \preceq$. Of course, the fact that $\preceq$ holds in this case. That is, for all possibly empty schemas $X$ and every tuple $y$, we have:

- If $y$ is a subtuple of $y_1$, $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_y(\delta)$, and in particular $\pi_X \sigma_\top(\delta) \preceq \pi_X \sigma_\top(\delta)$. Moreover, if $y \notin \pi_Y(\Delta)$ (which entails that $y_1 \notin \pi_Y(\Delta)$) we also have $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_y(\delta)$.

- If $X \rightarrow X_1$ is in $FD^+$ then $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_y(\delta)$, and in particular $\pi_X \sigma_\top(\delta) \preceq \pi_X \sigma_\top(\delta)$. Moreover, if $y \notin \pi_Y(\Delta)$ we also have $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_y(\delta)$.

- If $Y \rightarrow X$ is in $FD^+$, $\pi_Y \sigma_\top(\delta) \preceq \pi_X \sigma_y(\delta)$. In this case, we also have $\pi_X \sigma_y(\delta) \preceq \pi_Y \sigma_y(\delta)$ and $\pi_Y \sigma_y(\delta) \preceq \pi_X \sigma_y(\delta)$.

- $\pi_U \sigma_\top(\delta) \preceq \pi_X \sigma_y(\delta)$, meaning that, as is the case for $\preceq$, $\pi_U \sigma_\top(\delta)$ is the less specific query in $Q(U)$ with respect to $\preceq$.

- If $u$ is a tuple over $U$ and $y$ is a subtuple of $u$, $\pi_X \sigma_y(\delta) \preceq \pi_X \sigma_u(\delta)$. Moreover, if $y \notin \pi_Y(\Delta)$ (which entails that $u \notin \Delta$) we also have $\pi_X \sigma_u(\delta) \preceq \pi_X \sigma_y(\delta)$.

Of course, the fact that $\preceq$ allows for more comparisons in $\Delta$ than $\preceq$ is relevant only if the support measure can be shown to be anti-monotonic with respect to $\preceq$. This is precisely what is stated in the following proposition.

**Proposition 4.3.** Let $\Delta$ be a table in $inst_{FD}(U)$. For all queries $q$ and $q_1$ in $Q(U)$, if $q \preceq q_1$ then $sup_\Delta(q_1) \leq sup_\Delta(q)$.
Proof:
Let \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X \sigma_{y_1}(\delta) \) be two queries in \( Q(U) \). We first note that the result is trivial if \( q_1 \notin Q(\Delta) \). So let us assume that \( q_1 \in Q(\Delta) \), and thus that \( q \in Q(\Delta) \). If \( Y_1 \rightarrow X_1 \) is in \( FD^+ \), then \( sup(q_1) = 1 \) and \( sup(q) \geq 1 \), thus the result also holds in this case.

We now consider the case where \( q \) and \( q_1 \) are in \( Q(\Delta) \) and where \( Y_1 \rightarrow X_1 \) is not in \( FD^+ \). If \( y_1 = \top \), as we assume \( q \preceq_{\Delta} q_1 \), \( Y_1 \rightarrow Y \) is in \( FD^+ \), and so, \( Y = \emptyset \). Thus, \( y = \top \) and the result directly follows from Lemma 3.2(1). If \( y_1 \neq \top \), the table \( \sigma_{y_1}(\Delta) \) satisfies \( X \rightarrow Y_1 \), and as \( XY_1 \rightarrow X_1 \) is in \( FD^+ \), \( \sigma_{y_1}(\Delta) \) satisfies \( X \rightarrow X_1 \). Thus, by Lemma 3.2(1), \( |\pi_X \sigma_{y_1}(\Delta)| \leq |\pi_X \sigma_{y_1}(\Delta)| \). As \( Y_1 \rightarrow Y \in FD^+ \) and \( y = \Delta_Y(1) \), by Lemma 3.2(2), we have \( |\pi_X \sigma_{y_1}(\Delta)| \leq |\pi_X \sigma_{y}(\Delta)| \). Therefore, we obtain \( |\text{ans}_\Delta(q_1)| \leq |\text{ans}_\Delta(q)| \), and the proof is complete.

However, contrary to the pre-ordering \( \preceq \), the pre-ordering \( \preceq_{\Delta} \) does not allow for a characterization of the anti-monotonicity of the support as general as in Theorem 4.1. In particular, it is not true that

\[
\text{If, for a given } \Delta \text{ in } inst_{FD}(U), \text{ we have } q \preceq_{\Delta} q_1 \text{, then, for every } \Delta' \text{ in } inst_{FD}(U), \sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q) \text{ holds.}
\]

To see this, when considering the table \( \Delta \) of Figure 1, we have \( \pi_{U} \sigma_{\text{beer}}(\delta) \preceq_{\Delta} \pi_{U} \sigma_{\text{p2}}(\delta) \) (because \( \text{Pid} \rightarrow \text{Ptype} \in FD^+ \) and \( \Delta_{\text{Ptype}}(p_2) = \text{beer} \)). On the other hand, in the table \( \Delta' \) obtained from \( \Delta \) by replacing all occurrences of \( \text{beer} \) by \( \text{wine} \), we have \( \sup_{\Delta'}(\pi_{U} \sigma_{\text{beer}}(\delta)) = 0 \) and \( \sup_{\Delta'}(\pi_{U} \sigma_{\text{p2}}(\delta)) = 1 \). Thus, although \( \Delta' \) is in \( inst_{FD}(U) \), we have \( \sup_{\Delta'}(\pi_{U} \sigma_{\text{beer}}(\delta)) < \sup_{\Delta'}(\pi_{U} \sigma_{\text{p2}}(\delta)) \).

Nevertheless, \( \preceq_{\Delta} \) can be shown to characterize the anti-monotonicity of the support, provided that we consider only tables in \( inst_{FD}(U) \) in which \( q \) and \( q_1 \) have the same ‘status’ as in \( \Delta \). In other words, we show that:

\[
q \preceq_{\Delta} q_1 \text{ holds for a given } \Delta \text{ in } inst_{FD}(U) \text{ if and only if } \sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q) \text{ holds for every table } \Delta' \text{ in } inst_{FD}(U) \text{ in which } q \text{ and } q_1 \text{ have the same ‘status’ as in } \Delta.
\]

The tables in which \( q \) and \( q_1 \) have the same ‘status’ as in \( \Delta \) are defined as follows.

Definition 4.3. Let \( \Delta \) be in \( inst_{FD}(U) \) and let \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X \sigma_{y_1}(\delta) \) be in \( Q(U) \). The set of all tables in which \( q \) and \( q_1 \) have the same ‘status’ as in \( \Delta \), denoted by \( inst_{FD}(\Delta, q, q_1) \), is the set of all tables \( \Delta' \) in \( inst_{FD}(U) \) such that:

\[
\begin{align*}
q & \in Q(\Delta') \text{ if and only if } q \in Q(\Delta), \\
q_1 & \in Q(\Delta') \text{ if and only if } q_1 \in Q(\Delta), \text{ and} \\
\text{if } q \text{ and } q_1 \text{ are in } Q(\Delta) \text{ and } Y_1 \rightarrow Y \in FD^+, \text{ then } \Delta_Y(y_1) = \Delta_Y'(y_1).
\end{align*}
\]

It is easy to see from Definition 4.3 that, for every \( \Delta \) in \( inst_{FD}(U) \) and all queries \( q \) and \( q_1 \) in \( Q(U) \), \( \Delta \) is in \( inst_{FD}(\Delta, q, q_1) \), and for every \( \Delta' \) in \( inst_{FD}(\Delta, q, q_1) \), \( inst_{FD}(\Delta, q, q_1) = inst_{FD}(\Delta', q, q_1) \).

Moreover, it should be clear from Definition 4.3 that testing whether \( \Delta' \) belongs to \( inst_{FD}(\Delta, q, q_1) \) is linear in the sum of the sizes of \( \Delta \) and \( \Delta' \).

On the other hand, the following proposition is an immediate consequence of Definition 4.2 and Definition 4.3.
Proposition 4.4. Let $\Delta$ be in $\text{inst}_FD(U)$ and $q$ and $q_1$ in $Q(U)$. Then, for every $\Delta'$ in $\text{inst}_FD(\Delta, q, q_1)$, $q \preceq \Delta q_1$ holds if and only if $q \preceq \Delta' q_1$ holds.

Referring back again to Example 1.1, let $\Delta'$ be the table obtained from $\Delta$ by replacing all occurrences of beer by wine. Then, it should be clear that for $q = \pi_U \sigma_{p_2}(\delta)$ and $q_1 = \pi_U \sigma_{\text{beer}}(\delta)$, $\Delta'$ is not in $\text{inst}_FD(\Delta, q, q_1)$. However, if we consider $q' = \pi_U \sigma_{p_1}(\delta)$ and $q'_1 = \pi_U \sigma_{\text{milk}}(\delta)$, then $\Delta'$ does belong to $\text{inst}_FD(\Delta, q', q'_1)$.

We now prove that $\preceq_\Delta$ characterizes anti-monotonicity of the support with respect to all tables in which the queries under comparison have the same ‘status’ as in $\Delta$. To do so, we show the following two lemmas.

**Lemma 4.1.** Let $X$, $Y$, $X_1$ and $Y_1$ attribute sets such that

- $Y_1 \rightarrow X_1 \not\in FD^+$ and
- $XY_1 \rightarrow X_1 \not\in FD^+$ or $Y_1 \rightarrow Y \not\in FD^+$.

Let $y$ and $y_1$ be tuples over $Y$ and $Y_1$, respectively, and $Y_m = \text{match}(y, y_1)$.

If $(Y_m^+ \setminus Y_m) \cap (Y \cap Y_1) = \emptyset$, then there exists $\Delta$ in $\text{inst}_FD(U)$ such that $q = \pi_X \sigma_y(\delta)$ and $q_1 = \pi_{X_1} \sigma_{y_1}(\delta)$ are in $Q(\Delta)$ and $\sup_\Delta(q) < \sup_\Delta(q_1)$.

**Proof:**

We first note that, under the hypotheses of the lemma, as $Y_1 \rightarrow X_1$ is not in $FD^+$, $X_1 \not\in \emptyset$. Given $y$ and $y_1$ over $Y$ and $Y_1$, respectively, in order to prove the lemma, we successively consider the following two cases: (1) $Y_m = Y \cap Y_1$ and (2) $Y_m \subset Y \cap Y_1$.

(1) If $Y_m = Y \cap Y_1$, we first notice that the condition $(Y_m^+ \setminus Y_m) \cap (Y \cap Y_1) = \emptyset$ is trivially satisfied.

In this case, we construct a two tuple table $\Delta = \{t, t_1\}$ of $\text{inst}_FD(U)$ such that (i) $t.Y = y$, (ii) $t.Y_1 = t_1.Y_1 = y_1$, and (iii) $\sup_\Delta(q) = 1$ and $\sup_\Delta(q_1) = 2$. To this end, we study separately the cases whereby $Y_1 \rightarrow Y$ is or not in $FD^+$.

(1.a) If $Y_1 \rightarrow Y \in FD^+$, the second item in the lemma implies that $XY_1 \rightarrow X_1$ is not in $FD^+$.

In other words, we have $Y^+ \subseteq Y_1^+$ and $X_1^+ \not\subseteq (XY_1)^+$. In this case, $(XY_1)^+ \not= U$ because, as $(XY_1)^+ = (XY_1)^+$ and $(XY_1)^+ = U$ implies that $(XY_1)^+ = U$, and thus that $X_1^+ \subseteq (XY_1)^+$, which is not possible. Now, let $\Delta = \{t, t_1\}$ where $t$ and $t_1$ are two tuples over $U$ such that:

- $t.(XY_1)^+ = t_1.(XY_1)^+$,
- $t.Y = t_1.Y = y$,
- $t.Y_1 = t_1.Y_1 = y_1$, and
- $t.Y_1 = t_1.Y_1 = y_1$.

We note that, for $t$ and $t_1$ to be defined, we must have $y.(Y \cap Y_1) = y_1.(Y \cap Y_1)$, which holds because $Y_m = Y_1 \cap Y_2$. Moreover, as $\text{match}(t, t_1) = (XY_1)^+$, by Proposition 3.1, $\Delta$ is in $\text{inst}_FD(U)$. It is also important to note that, in this case, we have $\Delta(Y_1) = y$, since $t.Y_1 = t_1.Y_1 = y_1$.

On the other hand, $\Delta$ clearly satisfies (i) and (ii) above, that is $t.Y = y$ and $t.Y_1 = t_1.Y_1 = y_1$. Regarding (iii), we have $\sup_\Delta(q) = 1$ (because $t.X = t_1.X$ and $t.Y = t_1.Y = y$), and $\sup_\Delta(q_1) = 2$ (because $t.Y_1 = t_1.Y_1 = y_1$ and, as $X_1 \not\subseteq (XY_1)^+$, $t.X_1 \not= t_1.X_1$). Therefore, $\sup_\Delta(q) < \sup_\Delta(q_1)$.

(1.b) If $Y_1 \rightarrow Y \not\in FD^+$, we have $Y^+ \not\subseteq Y_1^+$, and so, $Y_1^+ \not= U$. Let $\Delta = \{t, t_1\}$ where $t$ and $t_1$ are tuples over $U$ such that:
- $t.Y_1^+ = t_1.Y_1^+$,
- $t.Y = y$,
- $t.Y_1 = t_1.Y_1 = y_1$, and
- for every $A \notin Y_1^+$ (such attributes exist since $Y_1^+ \neq U$), $t.A \neq t_1.A$.

We note that, for $t$ and $t_1$ to be defined, we must have $y.(Y \cap Y_1) = y_1.(Y \cap Y_1)$, which holds because $Y_m = Y \cap Y_1$. Moreover, as $\text{match}(t, t_1) = Y_1^+$, by Proposition 3.1, $\Delta$ is in $\text{inst}_{FD}(U)$.

On the other hand, $\Delta$ clearly satisfies (i) and (ii) above, that is $t.Y = y$ and $t.Y_1 = t_1.Y_1 = y_1$. Regarding (iii), we have $t.Y \neq t_1.Y$, because assuming that $t.Y = t_1.Y$ implies that $Y \subseteq Y_1^+$ and thus that $Y^+ \subseteq Y_1^+$, which is not possible. Thus, $\sup_{\Delta}(q) = 1$. Moreover, as $X_1^+ \nsubseteq Y_1^+$, we have $t.X_1 \neq t_1.X_1$, and since $t.Y_1 = t_1.Y_1 = y_1$, we obtain $\sup_{\Delta}(q_1) = 2$. Therefore, $\sup_{\Delta}(q) < \sup_{\Delta}(q_1)$, which completes the proof of case (1).

(2) Let us now assume that $Y_m \subset Y \cap Y_1$. We notice that, in this case, given $\Delta$ over $U$, for $q$ and $q_1$ to be in $Q(\Delta)$, $y$ and $y_1$ must respectively belong to $\pi_Y(\Delta)$ and $\pi_{Y_1}(\Delta)$. Thus, $y.(Y \cap Y_1)$ and $y_1.(Y \cap Y_1)$ are two distinct tuples that must both belong to $\pi_{(Y \cap Y_1)}(\Delta)$. Consequently, if $\Delta$ is assumed to be $\text{inst}_{FD}(U)$, these two tuples must at least satisfy the functional dependencies of $FD^+$ concerning $Y \cap Y_1$, meaning that $FD^+$ should not contain functional dependencies $X \rightarrow A$ such that $X \subseteq Y_m$ and $A \in (Y \cap Y_1) \setminus Y_m$. Therefore, $(Y_m^+ \setminus Y_m) \cap (Y \cap Y_1)$ must be empty, and this is precisely what is assumed in the lemma.

On the other hand, as the inclusion $Y_m \subset Y \cap Y_1$ is strict, we have $Y \cap Y_1 \neq \emptyset$. Let $y'$ be a tuple over $Y$ such that:
- for every $A \in (Y \setminus ((Y \cap Y_1)Y_1^+))$, $y'.A \neq y.A$,
- for every $A \in Y \cap Y_1^+$, $y'.A = y.A$, and
- for every $A \in (Y \cap Y_1) \setminus Y_m$, $y'.A = y_1.A$.

We first notice that $y'$ is well defined, because $Y$ can be partitioned into $(Y \setminus ((Y \cap Y_1)Y_1^+))$, $Y \cap Y_1^+$ and $((Y \cap Y_1) \setminus Y_m)$ and, as we assume that $(Y_m^+ \setminus Y_m) \cap (Y \cap Y_1) = \emptyset$, $((Y \cap Y_1) \setminus Y_m) = ((Y \cap Y_1) \setminus Y_m)$.

Moreover, we have $\text{match}(y', y_1) = Y \cap Y_1$. Thus, denoting by $q'$ the query $\pi_X \sigma_{y'}(\delta)$, the proof of case (1) above shows that there exists $\Delta' = \{t, t_1\}$ in $\text{inst}_{FD}(U)$ such that $t.Y = y'$, $t_1.Y_1 = y_1$, $\sup_{\Delta'}(q_1) = 2$ and $\sup_{\Delta'}(q') = 1$. Now, let $t_2$ be a tuple over $U$ such that:
- $t_2.Y = y$,
- $t_2.Y_m^+ = t_1.Y_m^+$, and
- for every $A \notin Y_m^+$, $t_2.A \neq t.A$ and $t_2.A \neq t_1.A$.

We first note that $Y_m^+ \neq U$. Indeed, as $Y_m \subset Y \cap Y_1$ implies that $Y_m^+ \subseteq (Y \cap Y_1)^+$ and as $(Y \cap Y_1)^+ \subseteq Y^+ \cap Y_1^+$ always holds, we have $Y_m^+ \subseteq Y^+ \cap Y_1^+$. Therefore, $Y_m^+ = U$ implies that $Y^+ = Y_1^+ = U$ and thus that, in particular, $X_1^+ \subseteq Y_m^+$. As we assume that $Y_1 \rightarrow X_1$ is not in $FD^+$, this is not possible. Therefore, $t_2 \neq t$ and $t_2 \neq t_1$. As in the proof of case (1) above, $t$ and $t_1$ have been shown to be distinct, $t, t_1$ and $t_2$ are three pairwise distinct tuples.

Moreover, for $t_2$ to be defined, we must have:
- $y'.(Y_m^+ \cap Y) = y.(Y_m^+ \cap Y)$, because $t.(Y_m^+ \cap Y) = t_2.(Y_m^+ \cap Y) = y.(Y_m^+ \cap Y)$ and $t.(Y_m^+ \cap Y) = y'.(Y_m^+ \cap Y)$. This holds by definition of $y'$.
- For every $A$ in $Y \setminus Y_m^+$, $y'.A \neq y.A$, because in this case, $t.A \neq t_2.A$, $t.A = y'.A$ and $t_2.A = y.A$.

Indeed, let $A \in Y \setminus Y_m^+$. If $A \in Y_1$, then $A \in (Y \cap Y_1) \setminus Y_m^+$. Thus, $A \in (Y \cap Y_1) \setminus Y_m$, and so, by definition of $y'$, $y'.A = y_1.A \neq y.A$. If $A \notin Y_1$, then $A \in (Y \setminus ((Y \cap Y_1)Y_m^+))$, in which case, by definition of $y'$, $y'.A \neq y.A$. 

Lemma 4.2. For every $A$ in $(Y_1 \cap Y) \setminus Y_m^+$, $y_1.A \neq y.A$, because in this case, $t_1.A \neq t_2.A$, $t_2.A = y.A$ and $t_1.A = y_1.A$.

Indeed, let $A \in (Y_1 \cap Y) \setminus Y_m^+$. Then $A \in (Y \cap Y_1) \setminus Y_m$, and so, as above, $y'.A = y_1.A \neq y.A$. Thus, $y_1.(Y_1 \cap Y) \setminus Y_m^+ \neq y.((Y_1 \cap Y) \setminus Y_m^+)$. 

Now, let $\Delta = \Delta' \cup \{t_2\}$ and let us prove that $\Delta$ is in $\text{inst}_{FD}(U)$. Indeed, as $\Delta'$ satisfies $FD$, we have $\text{match}(t, t_1) = (\text{match}(t, t_1))'$, and by definition of $t_2$, we also have $\text{match}(t_2, t) = Y_m^+$. Moreover, in $\Delta'$, the proof of case (1) above shows that $t.Y_1 = t_1.Y_1$, and thus, $t.Y_m = t_1.Y_m$, which entails that $t.Y_m^+ = t_1.Y_m^+$. Therefore, $t_2.Y_m^+ = t.Y_m^+ = t_1.Y_m^+$, and so, $Y_m^+ \subseteq \text{match}(t_2, t_1)$. Since we have $\text{match}(t_2, t_1) \subseteq Y_m^+$ by construction of $t_2$, we obtain $\text{match}(t_2, t_1) = Y_m^+$. Hence, by Proposition 3.1, $\Delta$ is in $\text{inst}_{FD}(U)$.

Regarding the supports of $q$ and $q_1$ in $\Delta$, we now note that $t_2$ is the only tuple of $\Delta$ such that $t_2.Y = y$. Indeed, as $t.Y = y'$ and $y \neq y'$, we have $t.Y \neq y$. On the other hand, we have $t_1.(Y \cap Y_1) = y_1.(Y \cap Y_1) = y'.(Y \cap Y_1)$ and $y.(Y \cap Y_1) \neq y_1.(Y \cap Y_1)$ (as $Y \cap Y_1 \neq \emptyset$ and as we assume that $Y_m \neq Y \cap Y_1$). Since assuming that $t_1.Y = y$ entails that $t_1.(Y \cap Y_1) = y.(Y \cap Y_1)$, we have a contradiction showing that $t_1.Y \neq y$.

Therefore, we have $\sup_{\Delta}(q) = 1$. On the other hand, as $\Delta' \subseteq \Delta$, we have $\sup_{\Delta}(q_1) \leq \sup_{\Delta}(q)$, and thus $\sup_{\Delta}(q_1) \geq 2$ (because $\sup_{\Delta}(q_1) = 2$). Hence, we obtain that $\sup_{\Delta}(q_1) > \sup_{\Delta}(q)$, which completes the proof.

Lemma 4.2. Let $\Delta$ be in $\text{inst}_{FD}(U)$, and $q = \pi_X \pi_g(\delta)$ and $q_1 = \pi_X \pi_{g_1}(\delta)$ in $Q(\Delta)$ such that $Y_1 \rightarrow X_1 \notin FD^+$, $Y_1 \rightarrow Y \notin FD^+$, and $y \neq \Delta_Y(y_1)$. Then there exists $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$ such that $\sup_{\Delta}(q_1) > \sup_{\Delta}(q)$.

Proof:
Let $y' = \Delta_Y(y_1)$ and $y_m = \text{match}(y, y')$. Under the hypotheses of the lemma, we have $Y^+ \subseteq Y_1^+$ and $Y_m \subseteq Y$ (as assuming that $Y_m = Y$ entails that $y = \Delta_Y(y_1)$). Thus, $Y_m^+ \subseteq Y^+ \subseteq Y_1^+$. Moreover, we also have that $Y_1^+ \neq U$ (because $Y_1^+ = U$ entails that $Y_1 \rightarrow X_1 \in FD^+$).

Let us consider the table $\Delta' = \{t, t_1, t_1'\}$ where $t$, $t_1$ and $t_1'$ are tuples over $U$ defined as follows:

- $t.Y = y$,
- $t_1.Y = t_1'.Y = y'$,
- $t_1.Y_1 = t_1'.Y_1 = y_1$,
- $t.Y_m^+ = t_1'.Y_m^+ = t_1.Y_m^+$,
- $t_1.Y_1 = t_1'.Y_1$,
- for every $A$ not in $Y_m^+$, $t_1.A \neq t_1'.A, t_1.A \neq t.A$ and $t_1'.A \neq t.A$.

We note that, for $t_1$ and $t_1'$ to be defined, we must have $y_1.(Y \cap Y_1) = y'.(Y \cap Y_1)$, which holds because $Y \rightarrow Y_1$ is in $FD^+$ and $y' = \Delta_Y(y_1)$. Moreover, since $Y_1^+ \neq U$, the tuples $t$, $t_1$ and $t_1'$ are pairwise distinct. On the other hand, using Proposition 3.1, $\Delta'$ satisfies $FD$ because we have $\text{match}(t_1, t_1') = (\text{match}(t_1, t_1'))' = Y_1^+$, $\text{match}(t, t_1) = (\text{match}(t, t_1))' = Y_m^+$ and $\text{match}(t, t_1') = (\text{match}(t, t_1'))'$.

Moreover, we have $\Delta_Y(y_1) = \Delta_Y(y_1) = y'$, because $t_1.Y_1 = t_1'.Y_1 = y_1$ and $t_1.Y = t_1'.Y = y'$. Thus, as $q$ and $q_1$ are in $Q(\Delta)$ and in $Q(\Delta')$, by Definition 4.3, $\Delta' \in \text{inst}_{FD}(\Delta, q, q_1)$.

Regarding the supports of $q$ and $q_1$ in $\Delta'$, since $Y_m \subseteq Y_1.Y$ and $t_1'.Y$ are different than $y$, and thus, $\sup_{\Delta}(q_1) = 1$. On the other hand, as $Y_1 \rightarrow X_1 \notin FD^+$, we have $X_1^+ \notin Y_1^+$, and so $t_1.X_1 \neq t_1'.X_1$. Hence, $\sup_{\Delta}(q_1) \geq 2$, showing that $\sup_{\Delta}(q_1) > \sup_{\Delta}(q)$. Thus, the proof is complete.
Based on Lemma 4.1, Lemma 4.2, Proposition 4.3 and Proposition 4.4, we are now ready to prove the following theorem, which shows that the pre-ordering $\preceq_\Delta$ characterizes anti-monotonicity of the support with respect to all tables of $\text{inst}_\Delta$ in which the queries under comparison have the same ‘status’ as in $\Delta$.

**Theorem 4.2.** Let $\Delta$ be in $\text{inst}_{FD}(U)$ and $q$ and $q_1$ in $Q(U)$. Then, $q \preceq_\Delta q_1$ holds if and only if, for every $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$, $\sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q)$.

**Proof:**

The ‘only if’ part of the proof is a consequence of Proposition 4.3 and Proposition 4.4. Indeed, for every table $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$, by Proposition 4.4, $q \preceq_\Delta q_1$ implies that $q \preceq_{\Delta'} q_1$ and so, by Proposition 4.3, $\sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q)$.

Let us now turn to the ‘if’ part of the proof that can be stated as follows, by contraposition: For all $q = \pi_X\sigma_y(\delta)$ and $q_1 = \pi_X\sigma_{y_1}(\delta)$ in $Q(U)$, if $q \not\preceq_\Delta q_1$ then there exists $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$ such that $\sup_{\Delta'}(q_1) > \sup_{\Delta'}(q)$.

By Definition 4.2, assuming that $q \not\preceq_\Delta q_1$ implies that $q_1 \in Q(\Delta)$, $Y_1 \rightarrow X_1 \not\in FD^+$ and one of the following statement holds: (i) $XY_1 \rightarrow X_1 \not\in FD^+$, or (ii) $Y_1 \rightarrow Y \not\in FD^+$, or (iii) $Y_1 \rightarrow Y \in FD^+$ and $y \not\in \Delta_Y(y_1)$. We consider successively the two cases whereby $q$ is or not in $Q(\Delta)$.

- If $q \not\in Q(\Delta)$, then, for every $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$, we have $q \not\in Q(\Delta')$ and $q_1 \in Q(\Delta')$. Thus, $\sup_{\Delta'}(q_1) = 0$ and $\sup_{\Delta'}(q_1) \geq 1$. Hence, we have $\sup_{\Delta'}(q_1) > \sup_{\Delta'}(q)$.

- If $q \in Q(\Delta)$, then $q$ and $q_1$ are in $Q(\Delta)$. In this case, as mentioned in the proof of Lemma 4.1, denoting by $Y_m$ the attribute set $\text{match}(y, y_1)$, the fact that $\Delta \in \text{inst}_{FD}(U)$ implies that $(Y^+_m \setminus Y_m) \cap (Y \cap Y_1) = \emptyset$. Moreover, as we assume that $q \not\preceq_\Delta q_1$, $Y_1 \rightarrow X_1$ is not in $FD^+$ (see case 2 of Definition 4.2).

Now, if (iii) holds, then Lemma 4.2 applies, showing that there exists $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$ such that $\sup_{\Delta'}(q_1) > \sup_{\Delta'}(q)$, which completes the proof in this case.

On the other hand, if (iii) does not hold, then (i) or (ii) hold. In this case, Lemma 4.1 shows that there exists $\Delta'$ in $\text{inst}_{FD}(U)$ such that $q$ and $q_1$ are in $Q(\Delta')$ and $\sup_{\Delta'}(q_1) > \sup_{\Delta'}(q)$.

We now prove that $\Delta'$ is in $\text{inst}_{FD}(\Delta, q, q_1)$. First, we have that $q$ and $q_1$ are both in $Q(\Delta)$ and in $Q(\Delta')$. Thus, the first two items of Definition 4.3 are satisfied. Therefore, if $Y_1 \rightarrow Y$ is not in $FD^+$, we have $\Delta' \in \text{inst}_{FD}(\Delta, q, q_1)$.

If $Y_1 \rightarrow Y$ is in $FD^+$ then $\Delta_Y(y_1) = y$ (because (iii) is assumed not to hold), which implies that $Y_m = Y \cap Y_1$. On the other hand, it has been seen in the proof of Lemma 4.1 that, when $Y_1 \rightarrow Y$ is in $FD^+$ and $Y_m = Y \cap Y_1$ (i.e., in case (1.a) of the proof), we have $\Delta_Y(y_1) = y$. Therefore, if $Y_1 \rightarrow Y$ is in $FD^+$ then $\Delta_Y(y_1) = \Delta_Y(y_1)$. Consequently, by Definition 4.3, in this case again, we obtain that $\Delta'$ is in $\text{inst}_{FD}(\Delta, q, q_1)$.

Thus, in any case, we have shown that there exists $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$ such that $\sup_{\Delta'}(q_1) > \sup_{\Delta'}(q)$. Therefore, the proof is complete.

Referring back to our running example, we have seen in Example 4.3 that $q = \pi_{\text{Cid}}\sigma_{\text{beer}}(\delta)$ and $q_1 = \pi_{Q_{\text{beer}}}\sigma_{p_2}(\delta)$ are in $Q(\Delta)$ and $q \preceq_\Delta q_1$. Then, Theorem 4.2 states that this holds if and only if $\sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q)$ holds in every $\Delta'$ of $\text{inst}_{FD}(\Delta, q, q_1)$, i.e., in every $\Delta'$ of $\text{inst}_{FD}(U)$ such that $q$ and $q_1$ are in $Q(\Delta')$ and product $p_2$ is of type beer.

The following example shows that the conditions in Definition 4.3 are necessary for Theorem 4.2 to hold. That is, it is not true that $q \preceq_\Delta q_1$ holds if and only if for every $\Delta'$ in $\text{inst}_{FD}(U)$, $\sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q)$. 


Example 4.5. Let $U = \{A, B, C\}$ and $FD = \{B \rightarrow C\}$. Considering the tables $\Delta = \{abc\}$ and $\Delta' = \{ab'c'\}$ where $b \neq b'$ and $c \neq c'$ and the queries $q = \pi_{\Delta}\sigma_c(\delta)$ and $q_1 = \pi_{\Delta}\sigma_{C'}(\delta)$, then $\Delta$ and $\Delta'$ are in $\text{inst}_{FD}(U)$, $q \preceq \Delta q_1$, and we have:

- $q \in Q(\Delta)$, $q_1 \notin Q(\Delta)$ and $\text{sup}_\Delta(q_1) < \text{sup}_\Delta(q)$
- $q \notin Q(\Delta')$, $q_1 \in Q(\Delta')$ and $\text{sup}_{\Delta'}(q) < \text{sup}_{\Delta'}(q_1)$.

This shows that the first two conditions in Definition 4.3 are necessary in order to make sure that we consider tables in which the functions induced for $q$ and $q_1$ have the same status.

We now illustrate the necessity of the third condition as follows: let $\Delta = \{a'bc, ab'c\}$ and $\Delta' = \{abc, ab'c', a'b'c'\}$, where $a \neq a'$, $b \neq b'$ and $c \neq c'$. Then clearly, $\Delta$ and $\Delta'$ are in $\text{inst}_{FD}(U)$, $q \preceq \Delta q_1$, and:

- $q$ and $q_1$ are both in $Q(\Delta)$ and $Q(\Delta')$, but $\Delta C(b') \neq \Delta C'(b')$
- $\text{sup}_\Delta(q_1) < \text{sup}_\Delta(q)$ and $\text{sup}_{\Delta'}(q) < \text{sup}_{\Delta'}(q_1)$.

Thus, even if the first two conditions in Definition 4.3 hold, the third one must also hold, in order to make sure that we consider tables in which the functions induced for $y$ and $y_1$ by the functional dependencies are equal.

5. Equivalent Queries

5.1. Equivalence Relations

Each of the two pre-orderings $\preceq$ and $\preceq_\Delta$ induces an equivalence relation over $Q(U)$ defined as follows.

Definition 5.1. Let $q$ and $q_1$ be two queries in $Q(U)$. Then:

- $q$ and $q_1$ are said to be equivalent, denoted by $q \equiv q_1$, if $q \preceq q_1$ and $q_1 \preceq q$.
  
  The equivalence class of $q$ modulo $\equiv$ is denoted by $[q]$, and the set of all equivalence classes modulo $\equiv$ is denoted by $C(U)$.

- If $\Delta$ is in $\text{inst}_{FD}(U)$, $q$ and $q_1$ are said to be equivalent with respect to $\Delta$, denoted by $q \equiv_\Delta q_1$, if $q \preceq_\Delta q_1$ and $q_1 \preceq_\Delta q$.
  
  The equivalence class of $q$ modulo $\equiv_\Delta$ is denoted by $[q]_\Delta$, and the set of all equivalence classes modulo $\equiv_\Delta$ is denoted by $C_{\Delta}(U)$.

Two important remarks are in order regarding the definition of $\equiv_\Delta$ in Definition 5.1 above.

1. First, by Definition 4.1, it is easy to see that all queries $q$ in $Q(U) \setminus Q(\Delta)$ are equivalent modulo $\equiv_\Delta$. In the remainder of the article, the equivalence class $Q(U) \setminus Q(\Delta)$ will be denoted by $C^0_{\Delta}$.

2. Second, all queries $q = \pi_X\sigma_y(\delta)$ in $Q(\Delta)$ such that $Y \rightarrow X$ is in $FD^+$ are equivalent modulo $\equiv_\Delta$. Indeed, it is easy to see from Definition 4.1 that if $q$ and $q_1$ are such queries, then $q \equiv_\Delta q_1$. Conversely, if $q_1 = \pi_X\sigma_{y_1}(\delta)$ is in $[q]_\Delta$, then, according to Definition 5.1, $q \preceq_\Delta q_1$. Thus, it can be seen from the proof of Proposition 4.2 that this entails that $Y_1 \rightarrow X_1$ is in $FD^+$ (since $Y \rightarrow X \in FD^+$). In the remainder of the article, the equivalence class modulo $\equiv_\Delta$ containing all queries $q = \pi_X\sigma_y(\delta)$ of $Q(\Delta)$ such that $Y \rightarrow X$ is in $FD^+$ will be denoted by $C_{\Delta}$.
Moreover, as a consequence of Proposition 4.4, given \( \Delta \) in \( \text{inst}_{FD}(U) \) and two queries \( q \) and \( q_1 \) in \( \mathcal{Q}(U) \), we have 
\[ q_1 \Delta = q_1 \Delta \text{ and } q_1 \Delta = q_1 \Delta, \]
for every \( \Delta' \) in \( \text{inst}_{FD}(\Delta, q, q_1) \).

Each of the two pre-orderings \( \leq \) and \( \leq_{\Delta} \) over \( \mathcal{Q}(U) \) induces a partial ordering over \( \mathcal{C}(U) \) and \( \mathcal{C}_{\Delta}(U) \), respectively, also denoted by \( \leq \) and \( \leq_{\Delta} \), respectively. These orderings are defined as follows:

- For all classes \( [q] \) and \( [q_1] \) in \( \mathcal{C}(U) \), \( [q_1] \) is said to be more specific than \( [q] \), denoted by \( [q] \preceq [q_1] \), if \( q \preceq q_1 \) holds.

- If \( \Delta \) is in \( \text{inst}_{FD}(U) \), for all classes \( [q]_{\Delta} \) and \( [q_1] \Delta \) in \( \mathcal{C}_{\Delta}(U) \), \( [q_1] \Delta \) is said to be more specific than \( [q] \Delta \) with respect to \( \Delta \), denoted by \( [q] \Delta \preceq [q_1] \Delta \), if \( q \preceq q_1 \) holds.

It is easy to see that \( \preceq \) and \( \preceq_{\Delta} \) over \( \mathcal{C}(U) \) and \( \mathcal{C}_{\Delta}(U) \), respectively, are indeed two partial orderings that are independent from the chosen representatives in \( [q] \) and \( [q_1] \), and in \( [q]_{\Delta} \) and \( [q_1] \Delta \), respectively. This explains why we use the same notations for the pre-orderings in \( \mathcal{Q}(U) \) and their associated orderings in \( \mathcal{C}(U) \) and \( \mathcal{C}_{\Delta}(U) \).

As a consequence of Theorem 4.1 and Theorem 4.2, the following corollary holds.

**Corollary 5.1.** For all \( q \) and \( q_1 \) in \( \mathcal{Q}(U) \):

1. \( q \equiv q_1 \) holds if and only if, for every \( \Delta \) in \( \text{inst}_{FD}(U) \), \( \text{sup}_{\Delta}(q) = \text{sup}_{\Delta}(q_1) \).

2. If \( \Delta \) is in \( \text{inst}_{FD}(U) \), \( q \equiv q_1 \) holds if and only if, for every \( \Delta' \) in \( \text{inst}_{FD}(\Delta, q, q_1) \), \( \text{sup}_{\Delta'}(q) = \text{sup}_{\Delta'}(q_1) \).

We recall from Example 4.1 that, in the context of our running example, \( \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \preceq \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \) and \( \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \preceq \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \) hold. Thus these queries are equivalent with respect to \( \equiv \), and Corollary 5.1 above shows that their supports are equal in every table \( \Delta \) of \( \text{inst}_{FD}(U) \).

Similarly, we recall from Example 1.1 that we have \( \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \preceq_{\Delta} \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \) and \( \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \preceq_{\Delta} \pi_{\text{Cid}}\sigma_{\gamma}(\delta) \). Thus, these queries are equivalent with respect to \( \equiv_{\Delta} \), and Corollary 5.1 above shows that their supports are equal in every table \( \Delta' \) of \( \text{inst}_{FD}(\Delta, \pi_{\text{Cid}}\sigma_{\gamma}(\delta), \pi_{\text{Cid}}\sigma_{\gamma}(\delta)) \).

Based on Corollary 5.1, given a class \( [q] \) in \( \mathcal{C}(U) \) (respectively \( [q]_{\Delta} \) in \( \mathcal{C}_{\Delta}(U) \)), we denote by \( \text{sup}_{\Delta}( [q] ) \) (respectively \( \text{sup}_{\Delta}( [q] ) \)) the support in \( \Delta \) of \( [q] \) (respectively of \( [q]_{\Delta} \), i.e., the support in \( \Delta \) of any query in \( [q] \) (respectively in \( [q]_{\Delta} \).

Thus, similarly to individual queries, given a support threshold \( \text{min-sup} \) and a class \( [q] \) in \( \mathcal{C}(U) \) (respectively \( [q]_{\Delta} \) in \( \mathcal{C}_{\Delta}(U) \)), \( [q] \) (respectively of \( [q]_{\Delta} \)) is said to be frequent if \( \text{sup}_{\Delta}( [q] ) \geq \text{min-sup} \) (respectively \( \text{sup}_{\Delta}( [q]_{\Delta} ) \geq \text{min-sup} \).

Recalling that \( C^0_{\Delta} \) is the class that contains all queries in \( \mathcal{Q}(U) \setminus \mathcal{Q}(\Delta) \), we have \( \text{sup}_{\Delta}( C^0_{\Delta} ) = 0 \). On the other hand, since \( C^1_{\Delta} \) is the set of all queries \( q = \pi_{X}\sigma_{\gamma}(\delta) \) in \( \mathcal{Q}(\Delta) \) such that \( Y \rightarrow X \) is in \( FD^+ \), we have \( \text{sup}_{\Delta}( C^1_{\Delta} ) = 1 \).

The following theorem states that the support of equivalence classes is anti-monotonic with respect to the partial orderings \( \leq \) and \( \leq_{\Delta} \) over equivalence classes.

**Theorem 5.1.** For all \( q \) and \( q_1 \) in \( \mathcal{Q}(U) \):

1. \( [q] \preceq [q_1] \) holds if and only if, for every \( \Delta \) in \( \text{inst}_{FD}(U) \), \( \text{sup}_{\Delta}( [q_1] ) \leq \text{sup}_{\Delta}( [q] ). \)
2. If $\Delta$ is in $\text{inst}_{FD}(U)$, $[q]_\Delta \preceq [q_1]_\Delta$ holds if and only if, for every $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$, $\text{sup}_{\Delta'}([q_1]_{\Delta'}) \leq \text{sup}_{\Delta'}([q]_{\Delta'})$.

Proof:
1. The result follows directly from Theorem 4.1 and Corollary 5.1(1).
2. Since $\Delta' \in \text{inst}_{FD}(\Delta, q, q_1)$, by Proposition 4.4, $[q]_\Delta = [q]_{\Delta'}$ and $|[q]_\Delta| = |[q]_{\Delta'}|$. Thus, we have $\text{sup}_{\Delta'}([q]_{\Delta}) = \text{sup}_{\Delta'}([q]_{\Delta'})$ and $\text{sup}_{\Delta'}([q_1]_{\Delta}) = \text{sup}_{\Delta'}([q_1]_{\Delta'})$, and the result follows from Theorem 4.2 and Corollary 5.1(2), which completes the proof.

It is important to note that, given a table $\Delta$ in $\text{inst}_{FD}(U)$, the impact of Corollary 5.1 and Theorem 5.1 on the computation of frequent queries is as follows:

1. Only one computation per equivalence class is necessary. Consequently, instead of considering individual queries of $Q(U)$ in algorithms, it is enough to consider one query per equivalence class of $C(\Delta) = C(\Delta)(U)$.

2. Frequent classes can be computed using the Apriori trick ([11]). In other words, as for individual queries, given a support threshold $\text{min-sup}$, if $\text{sup}_{\Delta'}([q]_\Delta) < \text{min-sup}$ (respectively $\text{sup}_{\Delta'}([q]_\Delta) < \text{min-sup}$), then for every class $[q]_\Delta$ such that $[q]_\Delta \preceq [q_1]_\Delta$ (respectively $[q]_\Delta \preceq [q_1]_\Delta$ such that $[q]_\Delta \preceq [q_1]_\Delta$), we have that $\text{sup}_{\Delta'}([q]_\Delta) < \text{min-sup}$ (respectively $\text{sup}_{\Delta'}([q_1]_\Delta) < \text{min-sup}$). Consequently, the support of $[q]_\Delta$ (respectively $[q_1]_\Delta$) has not to be computed.

We also point out that, since $\preceq_\Delta$ refines $\preceq$, given a table $\Delta$ in $\text{inst}_{FD}(U)$, equivalence classes modulo $\equiv$ are smaller (with respect to set inclusion) than those modulo $\equiv_\Delta$. In other words, for every query $q$ in $Q(U)$, we have $[q]_\Delta \subseteq [q]_\Delta$. This implies that, given $\Delta$ in $\text{inst}_{FD}(U)$, it is preferable to compute frequent classes modulo $\equiv_\Delta$, because the cardinality of $C(\Delta)$ is greater than that of $C(\Delta)(U)$. This is precisely what has been considered in our previous work [13].

However, computing frequent classes instead of individual frequent queries pays off only if equivalent queries can be easily characterized. The next section deals with this issue.

5.2. Equivalence Classes

In what follows, we characterize the content of equivalence classes modulo $\equiv$ and $\equiv_\Delta$, using the notion of key of a schema, as defined below.

**Definition 5.2.** Given a schema $X$, $K$ is said to be a key of $X$ if $K$ is a minimal (with respect to set inclusion) schema such that $K^+ = X^+$. The set of all keys of $X$ is denoted by $\text{keys}(X)$.

It is important to note that it is not the case that all keys of $X$ are subsets of $X$. However $\text{keys}(X)$ contains at least one key $K$ such that $K \subseteq X$ (this is so because the set $\{Y \mid Y^+ = X^+ \land Y \subseteq X\}$ is not empty, since it contains $X$).

In order to illustrate this remark, consider the universe $U = \{A, B, C\}$ and the set of functional dependencies $FD = \{A \rightarrow B, B \rightarrow A\}$. For $X = AC$, we have $X^+ = ABC$ and $\text{keys}(X) = \{A, B\}$. Thus, in this case, one of the two keys is a subset of $X$, whereas the other one is not.

The following proposition characterizes the content of classes in $C(U)$.
Proposition 5.1. For every \( q = \sigma_X \delta \) in \( Q(U) \), \([q]\) is the set of all queries \( q_1 = \sigma_X \delta \) where \( y_1 = y \) and there exists \( K \) in \( \text{keys}(XY) \) such that \( (K \setminus Y^+) \subseteq X_1 \subseteq (XY)^+ \).

Proof:
Let \( Q \) be the set of queries as defined in the proposition, and \( q_1 = \sigma_X \delta \) in \([q]\). It follows from Definition 4.1 that \( (XY)^+ = (X_1 Y_1)^+ \) and \( y_1 = y \). Thus, \( Y_1 = Y \), and as \( (XY)^+ = (X_1 Y)^+ \), by Definition 5.2, we have \( \text{keys}(XY) = \text{keys}(X_1 Y) \). Therefore, there exists \( K \subseteq \text{keys}(XY) \) such that \( K \subseteq X_1 Y \subseteq (XY)^+ \). Hence, \( (K \setminus Y^+) \subseteq K \subseteq X_1 Y \), which entails that \( (K \setminus Y^+) \subseteq X_1 \) (as \( (K \setminus Y^+) \cap Y = \emptyset \)). So, we have \( (K \setminus Y^+) \subseteq X_1 \subseteq (XY)^+ \), and thus, \( q_1 \in Q \), showing that \([q]\) is a set.

Conversely, let \( q_1 = \sigma_X \delta \) be in \( Q \). Then, \( y = y_1 \) and \( (K \setminus Y^+) \subseteq X_1 \subseteq (XY)^+ \). Thus, as \( Y \subseteq (XY)^+ \), we have \( (K \setminus Y^+) Y \subseteq X_1 Y \subseteq (XY)^+ \), and so, \( ((K \setminus Y^+) Y)^+ \subseteq (X_1 Y)^+ \subseteq (XY)^+ \).

In order to prove that \( (X_1 Y)^+ = (XY)^+ \), we show that \( (K \setminus Y^+) Y)^+ = (XY)^+ \).

Indeed, since \( K = (K \setminus Y^+) Y^+ \) and \( (K \setminus Y^+) Y^+ \subseteq ((K \setminus Y^+) Y)^+ \), we have \( K^+ \subseteq ((K \setminus Y^+) Y)^+ \).

Moreover, as \( K \subseteq XY \), \( (K \setminus Y^+) Y \subseteq (XY \setminus Y^+) \). Thus, \( (K \setminus Y^+) Y \subseteq (XY \setminus Y^+) Y \). Since \( (XY \setminus Y^+) Y \subseteq XY \), we obtain that \( (K \setminus Y^+) Y \subseteq (XY)^+ \).

Therefore, \( (K \setminus Y^+) Y)^+ = (XY)^+ \), showing that \( (X_1 Y)^+ = (XY)^+ \). As this entails that \( X_1 Y \rightarrow X \) and \( XY \rightarrow X \) are in \( FD^+ \), we obtain that \( \pi_X \sigma_Y (\delta) \in [q] \), and the proof is complete.

To illustrate Proposition 5.1, let us consider \( q = \sigma_{\text{Pid Ptype}} \sigma_{\text{Ptype}} \) in the context of our running example. In this case, we have \( Y^+ = \text{Pid Ptype} \) and \( \text{Pid Ptype} = U \), and thus, \([q]\) is the set of all queries \( \pi_X \sigma_Y (\delta) \) such that \( \text{Cid} \subseteq X_1 \subseteq U \).

As a consequence of Proposition 5.1 above, for every \( q \) in \( Q(U) \), \([q]\) contains exactly one representative \( \pi_X \sigma_Y (\delta) \) such that \( X^+ = X \) and \( Y \subseteq X \). Therefore, \( C(U) \) is isomorphic to the set of all queries \( q = \pi_X \sigma_Y (\delta) \) such that \( X^+ = X \) and \( Y \subseteq X \). In the remainder of the article, we identify every class of \( C(U) \) with this unique, particular representative.

In order to characterize the content of classes in \( C(U) \), we first define the notion of \text{keys of a query} as follows.

Definition 5.3. For every \( q = \sigma_X \sigma_Y (\delta) \) in \( Q(\Delta) \), the set of \text{keys of q}, denoted by \( \text{Keys}(q) \), is the set of all queries \( q_0 = \sigma_{\delta_0} \) in \( Q(\Delta) \) such that

\[
\begin{align*}
X_0 &= K_0 \setminus Y^+, \text{ where } K_0 \in \text{keys}(XY) \text{ and} \\
y_0 &= \Delta Y_0, \text{ where } Y_0 \in \text{keys}(Y).
\end{align*}
\]

We note that, in Definition 5.3, the tuple \( y_0 \) is correctly defined because, as \( Y_0 \in \text{keys}(Y) \), \( Y_0^+ = Y^+ \), and thus, \( Y \rightarrow Y_0 \) is in \( FD^+ \). Moreover, applying Lemma 3.1 shows that we also have \( y = \Delta Y(y_0) \).

On the other hand, it follows from Definition 5.3 that, for every \( q = \pi_X \sigma_Y (\delta) \) in \( Q(\Delta) \) such that \( Y \rightarrow X \) is in \( FD^+ \), \( \text{Keys}(q) = \{ \pi_{\delta_0} \mid y_0 \in \text{keys}(Y) \land y_0 = \Delta Y(y_0) \} \). Indeed, in this case, \( X^+ \subseteq Y^+ \) and thus, for every \( X_0 \in \text{keys}(XY) \), \( X_0 \subseteq (XY)^+ = Y^+ \). Hence, \( X_0 \setminus Y^+ = \emptyset \).

Example 5.1. In the context of Example 1.1, let \( q = \pi_{\text{Cid Cname}} \) as \( \text{keys}(\text{Cid Cname}) = \{ \text{Cid} \} \), we have \( \text{Keys}(q) = \{ \pi_{\text{Cid Cname}} \} \).
For \( q' = \pi_{\text{Cid Ptype Qty}} \sigma_{pZ} (\delta) \), we have \( \text{keys}(\text{Cid Ptype Qty Pid}) = \{ \text{Cid Pid} \} \) and \( \text{keys}(\text{Pid}) = \{ \text{Pid} \} \). Since \( \text{Pid}'' = \text{Pid} \text{Ptype} \), we obtain \( \text{Keys}(q') = \{ \pi_{\text{Cid Ptype} pZ} (\delta) \} \).

Now, for \( q'' = \pi_{\text{Ptype} pZ} (\delta) \), since \( \text{Pid} \rightarrow \text{Ptype} \) and \( \text{keys}(\text{Pid}) = \{ \text{Pid} \} \), we have \( \text{Keys}(q'') = \{ \pi_0 \sigma_{pZ} (\delta) \} \).

The following lemma shows that for every query \( q \) in \( Q(\Delta) \), every key of \( q \) is in \( [q]_{\Delta} \).

**Lemma 5.1.** For every \( \Delta \) in \( \text{inst}_{FD}(U) \) and every query \( q \) in \( Q(\Delta) \), every \( q_0 \) in \( \text{Keys}(q) \) is such that \( q_0 \equiv_{\Delta} q \).

**Proof:**
Let us first assume that \( q \) is in \( C_{1,\Delta} \). In this case, as mentioned above, for every \( q_0 = \pi_X \sigma_y (\delta) \) in \( \text{Keys}(q) \), we have \( X_0 = \emptyset \), which entails that \( q_0 \in C_{1,\Delta} \). Therefore, we have \( q_0 \equiv_{\Delta} q \).

Let us now assume that \( q = \pi_X \sigma_y (\delta) \) is not in \( C_{1,\Delta} \). Given \( q_0 = \pi_X \sigma_y (\delta) \) in \( \text{Keys}(q) \), we successively prove that \( q \equiv_{\Delta} q_0 \) and \( q_0 \equiv_{\Delta} q \), which then implies that \( q_0 \equiv_{\Delta} q \).

- To prove that \( q \equiv_{\Delta} q_0 \), we first show that \( X \rightarrow Y \) is in \( FD^+ \), or that \( X^+ \subseteq (XY)^+ \). Indeed, by Definition 5.3, we have \( X_0^+ = (K_0 \setminus Y^+)^+ \) and \( K_0^+ = (XY)^+ \) (as \( K_0 \subseteq \text{keys}(XY) \)). Therefore, we obtain that \( K_0^+ = (XY)^+ \). Moreover, as \( Y_0 \in \text{keys}(Y) \), we have \( Y_0^+ = Y^+ \) and thus, \( Y \rightarrow Y_0 \) is in \( FD^+ \) and \( K_0^+ = (XY)^+ \), and as \( X_0^+ \subseteq K_0^+ \), we obtain that \( X_0^+ \subseteq (XY)^+ \).

- Since by Definition 5.3, we also have \( y_0 = \Delta_{Y_0}(y) \), we have \( q_0 \equiv_{\Delta} q \) holds.

Now, the following proposition characterizes equivalent classes modulo \( \equiv_{\Delta} \).

**Proposition 5.2.** For every \( \Delta \) in \( \text{inst}_{FD}(U) \), we have:

1. \( C_{0,\Delta} = Q(U) \setminus Q(\Delta) \).
2. \( C_{1,\Delta} = \{ \pi_X \sigma_y (\delta) \in Q(\Delta) \mid X \subseteq Y^+ \} \).
3. For every \( q = \pi_X \sigma_y (\delta) \) in \( Q(\Delta) \) such that \( Y \rightarrow X \) is not in \( FD^+ \):

   \[
   [q]_\Delta = \{ \pi_X \sigma_y (\delta) \mid (\exists \pi_X \sigma_y (\delta) \in \text{Keys}(q)) ((X_0 \subseteq X_1 \subseteq (X_0 Y_0)^+) \land (Y_0 \subseteq Y_1 \subseteq Y_0^+) \land (y_1 = \Delta_{Y_1}(y_0))) \}.
   \]

**Proof:**
The first two items have been stated previously, after Corollary 5.1. Thus, we only prove item 3.

If \( [q]_\Delta \neq C_{1,\Delta} \), denoting by \( Q \) the set of queries as defined in the proposition, let \( q_1 = \pi_X \sigma_y (\delta) \) be in \( [q]_\Delta \). It follows from Definition 4.2 that \( (XY)^+ = (X_1 Y_1)^+, Y^+ = Y_1^+ \) and \( y_1 = \Delta_{Y_1}(y) \).

As in the proof of Proposition 5.1, it can be seen that there exists \( K \in \text{keys}(XY) \) such that \( K \subseteq X_1 Y_1 \subseteq (XY)^+ \). Similarly, there exists \( Y_0 \in \text{keys}(Y) \) such that \( Y_0 \subseteq Y_1 \subseteq Y^+ \) and \( y_0 = \Delta_{Y_0}(y) \).
Moreover, for \( X_0 = K \setminus Y_0^+ \), as \( X_0 \cap Y_0^+ = \emptyset \) and \( Y_1 \subseteq Y_0^+ \), \( X_0 \cap Y_1 = \emptyset \). Since \( X_0 \subseteq K \subseteq X_1Y_1 \), we have \( X_0 \subseteq X_1Y_1 \), and thus, \( X_0 \subseteq X_1 \). Therefore \( q_1 \in Q \).

Conversely, let \( q_1 \) be in \( Q \). As \( X_0 \subseteq X_1 \), \( X_1 \rightarrow X_0 \in FD^+ \), and as \( Y_0 \subseteq Y_1 \), \( Y_1 \rightarrow Y_0 \in FD^+ \). Therefore, \( X_1Y_1 \rightarrow X_0Y_0 \in FD^+ \). As \( X_1 \subseteq (X_0Y_0)^+ \) and \( Y_1 \subseteq Y_0^+ \), \( X_0Y_0 \rightarrow X_1Y_1 \) and \( Y_0 \rightarrow Y_1 \) are in \( FD^+ \). Thus, (a) \( X_0Y_1 \rightarrow X_1Y_1 \) and \( Y_0 \rightarrow Y_1 \) are in \( FD^+ \), and (b) \( X_1Y_0 \rightarrow X_0Y_0 \) and \( Y_1 \rightarrow Y_0 \) are in \( FD^+ \).

Moreover, since \( q_1 \in Q \), we have \( y_1 = \Delta Y_1(y_0) \), which with (a) above shows that \( q_0 \preceq \Delta q_1 \). On the other hand, as \( Y_0^+ = Y_1^+ \), by Lemma 3.1, \( y_1 = \Delta Y_2(y_0) \) entails that \( y_0 = \Delta Y_0(y_1) \), which with (b) above shows that \( q_1 \preceq \Delta q_0 \). Hence, \( q_1 \equiv \Delta q_0 \). Applying Lemma 5.1, we have \( q_0 \equiv \Delta q \), and so, \( q_1 \equiv \Delta q \). Thus, \( q_1 \in [q]_\Delta \), which completes the proof.

As a consequence of Proposition 5.2, for every \( q = \pi_X \sigma_y(\delta) \) in \( Q(\Delta) \) such that \([q]_\Delta \neq C^1_\Delta \), \([q]_\Delta \) contains exactly one representative \( \pi_X \sigma_y(\delta) \) such that \( X' = X^+ \), \( Y' = Y^+ \) and \( Y' \subset X' \).

Thus, \( C(\Delta)(U) \setminus \{C^0_\Delta, C^1_\Delta \} \) is isomorphic to the set of all queries \( \pi_X \sigma_y(\delta) \) in \( Q(\Delta) \) such that \( X^+ = X \), \( Y^+ = Y \) and \( Y \subset X \). In the remainder of the article, we identify every class of \( C(\Delta)(U) \setminus \{C^0_\Delta, C^1_\Delta \} \) with this unique, particular representative.

Considering again the queries \( q = \pi_{CidCname} \sigma\pi(\delta) \) and \( q' = \pi_{CidPtype} \sigma_{qty} \sigma_{p2}(\delta) \) of Example 5.1, we have:

\[
[q]_\Delta = \{\pi_{Cid\sigma\pi(\delta)}, \pi_{CidCname\sigma\pi(\delta)}, \pi_{CidCaddr\sigma\pi(\delta)}, \pi_{CidCnameCaddr\sigma\pi(\delta)}\}
\]

\[
[q']_\Delta = \{\pi_X \sigma_y(\delta) \mid Y_0 \subseteq U \land (y = q_2 \lor y = q_2 beer)\}.
\]

Thus the equivalence classes \([q]_\Delta \) and \([q']_\Delta \) are respectively represented by \( \pi_{CidCnameCaddr Pid Ptype} \sigma_{qty} \sigma_{p2} \sigma_{beer}(\delta) \) and \( \pi_{CidCnameCaddr Pid Ptype} \sigma_{qty} \sigma_{p2} \sigma_{beer}(\delta) \) (or more simply, by \( \sigma_{p2} \sigma_{beer}(\delta) \)).

We note that, for the query \( q'' = \pi_{Ptype} \sigma_{p2}(\delta) \) of Example 5.1, we have \([q'']_\Delta = C^1_\Delta \) because \( Pid \rightarrow Ptype \in FD^+ \) and \( p_2 \in \pi_{Pid}(\Delta) \). As a consequence, no particular representative of this class is considered.

The following example shows exhaustive comparisons of equivalence classes in the simple case of Example 4.2 and Example 4.4.

**Example 5.2.** As in Example 4.2, let us consider the case where \( U = \{A, B\} \) and \( FD = \emptyset \). Assuming that \( dom(A) = \{a\} \) and \( dom(B) = \{b\} \) and considering the particular representatives as defined above, the set \( C(U) \) along with comparisons according to \( \preceq \) are shown below (case (a)).

\[\begin{align*}
\pi_A \sigma_\pi(\delta) & \preceq \pi_B \sigma_\pi(\delta) \\
\pi_A \sigma_\beta(\delta) & \preceq \pi_B \sigma_\beta(\delta) \\
\pi_B \sigma_\beta(\delta) & \preceq \pi_A \sigma_\alpha(\delta) \\
\pi_A \sigma_{\alpha\beta}(\delta) & \preceq \pi_B \sigma_{\alpha\beta}(\delta)
\end{align*}\]

(a) The set \( C(U) \)

\[\begin{align*}
\pi_A \sigma_\pi(\delta) & \preceq \pi_B \sigma_\pi(\delta) \\
\pi_A \sigma_\beta(\delta) & \preceq \pi_B \sigma_\beta(\delta) \\
\pi_B \sigma_\beta(\delta) & \preceq \pi_A \sigma_\alpha(\delta) \\
\pi_A \sigma_{\alpha\beta}(\delta) & \preceq \pi_B \sigma_{\alpha\beta}(\delta)
\end{align*}\]

(b) The set \( C(\Delta)(U) \)
Now, considering $\Delta = \{ab\}$ as in Example 4.4, we note that, in this case $C_\Delta^0 = \emptyset$. Thus, the set $C_\Delta(U)$ along with comparisons according to $\preceq_\Delta$ are shown above (case (b)).

We draw attention on the fact that, even in this simple case without functional dependencies, the structures of $C(U)$ and $C_\Delta(U)$ are simpler than those of $Q(U)$ shown in Example 4.2 and Example 4.4, respectively.

We point out that, although the graphs in the previous example show that, in this case, $C(U)$ and $C_\Delta(U)$ are lattices, this does not hold in general. The following example, borrowed from [13], shows a case where neither $C(U)$ nor $C_\Delta(U)$ has a lattice structure.

**Example 5.3.** Let $U = \{A, B, C, D\}$ and $FD = \{ABC \rightarrow D\}$. In order to see that in this case, $C(U)$ is not a lattice, we consider the two classes represented by $q_1 = \pi_B \sigma_T(\delta)$ and $q_2 = \pi_{ABD} \sigma_{a}(\delta)$ where $a$ is in $\text{dom}(A)$, and we show that $[q_1]$ and $[q_2]$ have two distinct greatest lower bounds in $C(U)$.

Indeed, for $q = \pi_{BC} \sigma_T(\delta)$ and $q' = \pi_{BD} \sigma_T(\delta)$, for $i = 1, 2$, $[q] \preceq [q_i]$ and $[q'] \preceq [q_i]$, i.e.,

1. $\pi_{BC} \sigma_T(\delta) \preceq \pi_{BD} \sigma_T(\delta)$ and $\pi_{BC} \sigma_T(\delta) \preceq \pi_{ABD} \sigma_{a}(\delta)$,
2. $\pi_{BD} \sigma_T(\delta) \preceq \pi_{BD} \sigma_T(\delta)$ and $\pi_{BD} \sigma_T(\delta) \preceq \pi_{ABD} \sigma_{a}(\delta)$.

(These comparisons hold because $FD^+$ contains the following dependencies: $BC \rightarrow B$, $ABC \rightarrow ABD$, $BD \rightarrow B$ and $ABD \rightarrow ABD$.)

Thus, $[q]$ and $[q']$ are two distinct lower bounds of $[q_1]$ and $[q_2]$ in $C(U)$. Moreover, let $q_0 = \pi_{X_0 \sigma_{y_0}}(\delta)$ where $X_0 = X_0^+$ and $Y_0 \subseteq X_0$ be such that, for $i = 1, 2$, $[q] \preceq [q_0] \preceq [q_i]$. Then, we have $[q] \neq [q_0]$ and:

1. $X_0 \subseteq (BCY_0)^+$, $B \subseteq X_0$ and $ABD \subseteq (X_0A)^+$;
2. $A$ is a subtuple of $y_0$ and $y_0$ is a subtuple of $A$.

Then, by item 2 above, we obtain $y_0 = A$. So, $Y = \emptyset$, and thus, $(BCY_0)^+ = BC$. Taking into account that $[q] \neq [q_0]$, the first inclusion of item 1 above can be written as $X_0 \subset BC$. Therefore, either (i) $X_0 = \emptyset$ or (ii) $X_0 = B$ or (iii) $X_0 = C$. In case (i), the last inclusion of item 1 implies that $ABD \subseteq A$, which is not possible. Similarly, in cases (ii) and (iii), the last inclusion of item 1 implies that $ABD \subseteq AB$ and $ABD \subseteq AC$, respectively, which is again not possible.

As a consequence, $[q]$ is a greatest lower bound of $[q_1]$ and $[q_2]$ in $C(U)$. Since a similar reasoning holds when considering $[q']$ instead of $[q]$, we conclude that $[q]$ and $[q']$ are two distinct lower bounds of $[q_1]$ and $[q_2]$ in $C(U)$. Consequently, $C(U)$ is not a lattice in this case.

Now, considering $\Delta = \{abcd\}$, for the same reasons as above, we also have:

1. $\pi_{BC} \sigma_T(\delta) \preceq \pi_{BD} \sigma_T(\delta)$ and $\pi_{BC} \sigma_T(\delta) \preceq \pi_{ABD} \sigma_{a}(\delta)$,
2. $\pi_{BD} \sigma_T(\delta) \preceq \pi_{BD} \sigma_T(\delta)$ and $\pi_{BD} \sigma_T(\delta) \preceq \pi_{ABD} \sigma_{a}(\delta)$.

It follows that $[q]_{\Delta}$ and $[q']_{\Delta}$ are two distinct lower bounds of $[q_1]_{\Delta}$ and $[q_2]_{\Delta}$ in $C_\Delta(U)$. Moreover, for $q_0 = \pi_{X_0 \sigma_{y_0}}(\delta)$ where $X_0 = X_0^+$, $Y_0 = Y_0^+$ and $Y_0 \subseteq X_0$, if, for $i = 1, 2$, $[q] \preceq [q_0] \preceq [q_i]_{\Delta}$ then we have $[q]_{\Delta} \neq [q_0]_{\Delta}$ and:

1. $X_0 \subseteq (BCY_0)^+$, $B \subseteq X_0$ and $ABD \subseteq (X_0A)^+$;
2. $\emptyset \subseteq Y_0$, $Y_0 \subseteq \emptyset$, and $Y_0 \subseteq A$ along with $\Delta Y_0(a) = y_0$.

In this case, item 2 above entails that $Y_0 = \emptyset$, and thus that $y_0 = \top$, which satisfies $\Delta Y_0(a) = y_0$. Thus, as above, it can be shown that $q_0$ does not exist. As a consequence, the set $C_\Delta(U)$ is not a lattice.

6. Conclusion and Further Work

In this article, we have considered the problem of comparing conjunctive queries for the efficient computation of the supports of projection-selection queries on a given relational table satisfying a given set of functional dependencies. In this setting, we defined and characterized two pre-orderings with respect to which the support measure has been shown to be anti-monotonic. Furthermore, we showed that these pre-orderings allow to consider equivalence classes of queries instead of individual queries.

We are currently implementing algorithms for mining frequent projection-selection queries in the case of the pre-ordering $\preceq_\Delta$ defined in this article.

Regarding possible extensions of the present approach, we are investigating the following issues:

- Extending our approach to the case of projection-selection-join queries. We are investigating this issue under the standard restriction whereby joins are performed in the presence of key and foreign-key constraints. The particular case of star schemas has been considered in [14].

- Extending the selection conditions to equalities of the form $Y = Y'$ where $Y$ and $Y'$ are two schemas, as done in [10, 11], is another issue that we are investigating in the context of the present work.

- Finally, we intend to study the discovery of functional dependencies and conditional functional dependencies in the context of the present work.

References


