Comparing and Mining Conjunctive Queries from a Relational Table with Functional Dependencies

Tao-Yuan Jen
Université Cergy-Pontoise - ETIS - CNRS - F-95000 Cergy-Pontoise
tao-yuan.jen@u-cergy.fr

Dominique Laurent
Université Cergy-Pontoise - ETIS - CNRS - F-95000 Cergy-Pontoise
dominique.laurent@u-cergy.fr

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

Abstract. In this paper we study the problem of mining all frequent queries in a relational table, a problem known to be intractable even for conjunctive queries. We restrict our attention to projection-selection queries and we assume that the table to be mined satisfies a set of functional dependencies. Under these assumptions, we define 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 paper 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 optimized computation of frequent 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 paper, based on our previous work [14].

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

In this paper we study the problem of mining frequent conjunctive queries in a (relational) table $\Delta$ over an attribute set $U$, where a query is said to be frequent if the cardinality of its answer is above a given threshold.

This is an important but challenging problem of data mining, known to be intractable even for conjunctive queries [8]. Indeed, 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, 12, 17, 18, 15, 9, 8, 10, 11, 14], 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, borrowing 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 projection-selection queries. Given a relational table $\Delta$ in $\text{inst}_{FD}(U)$, a 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 of $\pi_Y(\Delta)$.

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.

Moreover, 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. The implications of these results are very important from a computational point of view. Indeed, as seen in [14]:

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.
Although this paper does not provide algorithms for the computation of frequent projection-selection queries, the work presented in this paper can be seen as a generalization of our previous work [14]. Indeed, in our previous work, we presented algorithms concerning one of the two pre-orderings defined in this paper, 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 \( \text{linear in } |U| \) (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 paper).

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

- \( \text{Cid}, \text{Cname} \) and \( \text{Caddr} \) stand for Customer Identifier, Customer Name and Customer Address,
- \( \text{Pid} \) and \( \text{Ptype} \) 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 1.1 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: \textit{Customer}(<\text{Cid}, \text{Cname}, \text{Caddr}>), \textit{Product}(<\text{Pid}, \text{Ptype}>), \textit{Sales}(<\text{Cid}, \text{Pid}, \text{Qty}>) where \textit{Customer} and \textit{Product} are dimension tables and \textit{Sales} is the fact table.

Assuming that all frequent projection-selection queries are computed using the table \( \Delta \), we argue that it is relevant to 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 shall come back later in the paper (see end of Section 3) on examples of more general association rules that allow to mine functional dependencies, as well as \textit{conditional functional dependencies} ([4]).

---

**Figure 1.** The table \( \Delta \) of the running example
In this paper, 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 1.1, it is easy to verify that we have \( |\pi_{\text{Cid}}(\Delta)| \geq |\pi_{\text{Caddr}}(\Delta)| \), and that this inequality holds because the table \( \Delta \) satisfies the dependency \( \text{Cid} \rightarrow \text{Caddr} \) (which is a consequence of \( \text{FD} \)). Notice that this equality holds in any table that satisfies the dependencies in \( \text{FD} \).

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

We point out that, although not equivalent according to standard query equivalence [19], the queries \( \pi_{\text{Cid}}(\Delta) \) and \( \pi_{\text{CidCaddr}}(\Delta) \) 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 \( |\sigma_{\text{Ptype}=\text{p}_2 \text{beer}}(\Delta)| \leq |\sigma_{\text{Pid}=\text{p}_2}(\Delta)| \), and that this inequality holds in any table because, as \( \text{p}_2 \) is a subtuple of \( \text{p}_2 \text{ beer} \), this is a case of query containment [19].

On the other hand, it is easy to verify that \( |\sigma_{\text{Pid}=\text{p}_2}(\Delta)| \leq |\sigma_{\text{Ptype}=\text{beer}}(\Delta)| \), and that this inequality holds because the table \( \Delta \) satisfies the dependency \( \text{Pid} \rightarrow \text{Ptype} \) and \( \text{p}_2 \) is associated with \( \text{beer} \). Clearly, this inequality holds in any table that satisfies the dependency \( \text{Pid} \rightarrow \text{Ptype} \) and in which the \( \text{Pid} \) value \( \text{p}_2 \) is associated with the \( \text{Ptype} \) value \( \text{beer} \).

Notice however that this last inequality does not hold in every table that satisfies the dependencies in \( \text{FD} \). Indeed, consider the table \( \Delta' \) obtained from \( \Delta \) by replacing all occurrences of \( \text{beer} \) by \( \text{wine} \). Then \( \Delta' \) still satisfies \( \text{FD} \) but now, \( \text{p}_2 \) is associated with \( \text{wine} \) instead of \( \text{beer} \). As a result, we have \( |\sigma_{\text{Pid}=\text{p}_2}(\Delta')| > |\sigma_{\text{Ptype}=\text{beer}}(\Delta')| \), since the latter answer set in \( \Delta' \) is empty.

As shown in the previous example, mining 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]), two major issues in data mining.

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 \( \text{FD} \) 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 \( \text{FD} \).

2. Given a table \( \Delta \) satisfying \( \text{FD} \), the second pre-ordering we consider, denoted by \( \preceq_{\Delta} \), refines 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 (to be defined later on). In other words, 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 \( \text{FD} \) and such that \( q \) and \( q' \) have the same status in \( \Delta' \) as in \( \Delta \).
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 paper 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 compare and characterize the content of equivalence classes modulo each of the equivalence relations. In Section 6, we show that the sets of equivalence classes modulo the two equivalence relations are not lattices in general, and we identify particular cases for which they have a lattice structure. Section 7 concludes the paper and discusses future work.

2. Related Work

The work in [8] considers conjunctive queries, as we do in this paper, 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 paper.

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 generalized in [11] to the case of projection-selection-join queries.

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 paper. However, in [10] queries are compared according to the standard notion of query equivalence ([19]), and in [11], queries are compared according to a partial ordering called diagonal containment.

Comparing our present contribution to the work in [10, 11], in our approach, we provide two pre-orderings for query comparison that are shown to generalize standard query equivalence and diagonal containment, while taking functional dependencies into account (which is not the case in [10, 11]).

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 ([19]). 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 [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, 17, 18] 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 [18], 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 [18], all these approaches are restricted to conjunctive queries, as is the case in the present paper.

To end this section, we emphasize that, to the best of our knowledge, this work along with that of [15, 14], is the first attempt to consider explicitly constraints on the data set, such as functional 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 [19]. 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, $\text{dom}(X) = \Pi_{A \in X}(\text{dom}(A))$.

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 $X$, $x.Y$ 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 [19], we denote by $FD^+$ the set of all functional dependencies that can be inferred from $FD$, based on Armstrong axioms ([2]). Given a table $\Delta$ in $\text{inst}_{FD}(U)$, for every $X \rightarrow Y$ in $FD^+$ and every tuple $x$ over $X$, we denote by $\Delta_Y(x)$ the tuple over $Y$ associated in $\Delta$ with $x$ through $X \rightarrow Y$.

Referring back to Figure 1.1, $Cid \rightarrow Caddr$ is in $FD^+$, and we have, for instance, $\Delta_{Caddr}(c_1) = \text{Paris}$, as all tuples $t$ such that $t.Cid = c_1$ satisfy $t.Caddr = \text{Paris}$.

In this paper, given an attribute set $X$, we consider the notion of closure of $X$ as in [19], 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.

Functional dependencies allow for comparisons of the cardinalities of the answers to queries, as stated in the following lemma.
Lemma 3.1. 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.

In the standard relational model, schemas are assumed to be nonempty sets. However, for the purposes of this paper, 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.

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 the empty table results in the 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 Armstrong axioms.
- For every $X$ (possibly empty), every table $\Delta$ satisfies $X \rightarrow \emptyset$. Thus, for every $x$ in $\text{dom}(X)$, $\top = \Delta_{\emptyset}(x)$.
- For every set of functional dependencies $FD$, $\emptyset^+ = \emptyset$.

We note however that, in this paper, 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 paper 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 $t$ and $t'$ over $U$, $\text{match}(t, t')$ denotes the set of all attributes $A$ of $U$ such that $t.A = t'.A$.

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., $\text{match}(t, t') = (\text{match}(t, t'))^\top$. 
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 \( y = \top \), 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 \( \text{inst}_{FD}(U) \) and a query \( q = \pi_X\sigma_y(\delta) \) in \( Q(U) \), the answer to \( q \) in \( \Delta \), denoted by \( \text{ans}_\Delta(q) \), 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 paper, 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 \( \pi_X\sigma(\delta) \), where \( X \) is a nonempty schema, are simply the relational queries of the form \( \pi_X(\delta) \). Consequently \( \pi_U\sigma(\delta) \) stands for the simple query \( \delta \).

Moreover, for every \( \Delta \) in \( \text{inst}_{FD}(U) \) and every \( q \) in \( Q(U) \setminus Q(\Delta) \), \( \text{ans}_\Delta(q) = \emptyset \). In particular, if \( y \in \pi_Y(\Delta) \) then \( \text{ans}_\Delta(\pi_0\sigma_y(\delta)) = \{ \top \} \), else \( \text{ans}_\Delta(\pi_0\sigma_y(\delta)) = \emptyset \).

Example 3.1. Referring back to Example 1.1, \( \pi_{\text{Cid}}\sigma(\delta) \) and \( \pi_{\text{Caddr}}\sigma_{\text{Paris beer}}(\delta) \) are queries in \( Q(\Delta) \), and we have: \( \text{ans}_\Delta(\pi_{\text{Cid}}\sigma(\delta)) = \{ c_1, c_2, c_3, c_4 \} \), and \( \text{ans}_\Delta(\pi_{\text{Caddr}}\sigma_{\text{Paris beer}}(\delta)) = \{ \text{Paris} \} \). Similarly, \( \pi_0\sigma_{\text{Paris beer}}(\delta) \) is also a query in \( Q(\Delta) \), and we have \( \text{ans}_\Delta(\pi_0\sigma_{\text{Paris beer}}(\delta)) = \{ \top \} \).

On the other hand, the queries \( \pi_{\text{Cid}}\sigma_{\text{NY milk}}(\delta) \) and \( \pi_0\sigma_{\text{NY milk}}(\delta) \) are in \( Q(U) \) but not in \( Q(\Delta) \), because \( \text{NY milk} \) is not in \( \pi_{\text{Caddr}}P_{\text{type}}(\Delta) \). Thus, \( \text{ans}_\Delta(\pi_{\text{Cid}}\sigma_{\text{NY milk}}(\delta)) = \text{ans}_\Delta(\pi_0\sigma_{\text{NY milk}}(\delta)) = \emptyset \).

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 \( \text{inst}_{FD}(U) \) and \( q \) be in \( Q(U) \). The support of \( q \) in \( \Delta \), denoted by \( \text{sup}_\Delta(q) \), is the cardinality of the answer to \( q \) in \( \Delta \), i.e., \( \text{sup}_\Delta(q) = |\text{ans}_\Delta(q)| \).

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

We notice that for every \( \Delta \) in \( \text{inst}_{FD}(U) \) and every query \( q \) in \( Q(U) \), we have \( 0 \leq \text{sup}_\Delta(q) \leq |\Delta| \).

Moreover, for every \( q \) in \( Q(U) \), we have that \( q \notin Q(\Delta) \) if and only if \( \text{sup}_\Delta(q) = 0 \).

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 \( \text{sup}_\Delta(q) = 0 \). Conversely, if \( q \in Q(\Delta) \), then \( \Delta \) contains at least one tuple \( t \) such that \( t.Y = y \), and thus, \( \text{sup}_\Delta(q) \geq 1 \).

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 (\cite{4}).

Indeed, given a table \( \Delta \), it can be seen that if in \( \Delta \), the confidence of the rule \( \pi_{X_1X_2}(\delta) \Rightarrow \pi_{X_1}(\delta) \) is 1, then \( \Delta \) satisfies the functional dependency \( X_1 \rightarrow X_2 \). This is so because, if the confidence of the rule \( \pi_{X_1X_2}(\delta) \Rightarrow \pi_{X_1}(\delta) \) is 1, then \( \sup_{\Delta}(\pi_{X_1X_2}(\delta)) = \sup_{\Delta}(\pi_{X_1}(\delta)) \), meaning that the answers in \( \Delta \) of \( \pi_{X_1X_2}(\delta) \) and \( \pi_{X_1}(\delta) \) have the same cardinality.

Therefore, this implies that, in the table \( \pi_{X_1X_2}(\Delta) \), 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 \cite{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 (\( i \)) \( \pi_{X_1X_2} \sigma_{y_1^\tau}(\delta) \Rightarrow \pi_{X_1} \sigma_{y_1^\tau}(\delta) \) and (\( ii \)) \( \sigma_{y_1^\tau}(\delta) \Rightarrow \sigma_{y_2^\tau}(\delta) \) are both equal to 1.

\section{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 \( \text{inst}_{FD}(U) \).

\subsection{Query Comparison characterizing Anti-monotonicity}

\textbf{Definition 4.1.} Let \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_{X_1} \sigma_{y_1}(\delta) \) 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:
Definition 4.1: \( XY \preceq q \) if \( q \) is a subtuple of \( XY \) (because \( XY_1 \rightarrow X \) is a trivial dependency). In particular, \( \pi_X \sigma_Y(\delta) \preceq \pi_X \sigma_{y_1}(\delta) \) (because \( XY_1 \rightarrow X \) 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, \( \pi_X \sigma_Y(\delta) \preceq \pi_X \sigma_Y(\delta) \) (because \( X \rightarrow \emptyset \) is assumed to be in any \( FD^+ \)).

If \( Y \rightarrow X \) is in \( FD^+ \), \( \pi_{y_1} \sigma_Y(\delta) \preceq \pi_X \sigma_Y(\delta) \) (because \( XY \rightarrow X \) is assumed to be in \( FD^+ \) and \( \pi_{y_1} \sigma_Y(\delta) \preceq \pi_X \sigma_Y(\delta) \) (because \( Y \rightarrow X \) is assumed to be in \( FD^+ \)).

\( \pi_{u \sigma_Y(\delta)} \preceq \pi_X \sigma_Y(\delta) \) (because \( \preceq \) 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 \( \pi_{u \sigma_Y(\delta)}(\delta) \) is the less specific query \( \pi_X \sigma_Y(\delta) \).

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 \( X_1 \subseteq U \) 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 \( X_1 Y_2 \rightarrow X_2 \) 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 \( Y_1 \subseteq Y_2, X_1 Y_2 \rightarrow X_1 Y_1 \) is in \( FD^+ \). Therefore, \( X_1 Y_2 \rightarrow X_1 Y_2 \) is in \( FD^+ \) (because \( X_1 Y_1 \rightarrow X_1 \in FD^+ \)), and so, \( Y_2 \rightarrow X_2 \) is in \( FD^+ \) (because \( X_1 Y_2 \rightarrow X_2 \in FD^+ \)). Hence, \( q \preceq q_2 \), and the proof is complete.

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

**Example 4.1.** Referring back to Example 1.1, \( q = \pi_{\text{Cid}\sigma_Y(\delta)} \) and \( q_1 = \pi_{\text{CidCaddr}\sigma_Y(\delta)} \) 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 \( \text{CidCaddr} \rightarrow \text{Cid} \) 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. Assuming only one value in the domains of \( A \) and \( B \), denoted by \( a \) and \( b \), respectively, \( Q(U) \) contains all queries \( \pi_X \sigma_Y(\delta) \) where \( X \subseteq \{\emptyset, A, B, AB\} \) and \( Y \subseteq \{\top, a, b, ab\} \).

The pre-ordering \( \preceq \) 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.
We note that replacing item 1 of Definition 4.1 by \( X \rightarrow X_1 \subseteq FD^+ \) 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 \( \preceq \), because \( XY_1 \rightarrow X_1 \subseteq FD^+ \) does not imply \( X \rightarrow X_1 \subseteq FD^+ \).

For instance, in the context of our running example, \( \pi_{\text{Cid}} \sigma_T(\delta) \) and \( \pi_{\text{Qty}} \sigma_{\text{p}_2}(\delta) \) would not be comparable, whereas \( \pi_{\text{Cid}} \sigma_T(\delta) \preceq \pi_{\text{Qty}} \sigma_{\text{p}_2}(\delta) \) holds because \( \text{Cid \text{Id}} \rightarrow \text{Qty} \in FD^+ \).

Moreover, it can be seen that the pre-ordering \( \preceq \) generalizes query containment ([19]) and diagonal containment ([11]). Indeed, denoting these pre-orderings by \( \subseteq \) and \( \subseteq^D \), respectively, given two queries \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X \sigma_{y_1}(\delta) \) in \( Q(U) \), in our context, we have

- \( q_1 \subseteq q \) 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], \( q_1 \subseteq^D q \) holds if and only if \( q_1 \subseteq \pi_X(q) \), 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 \( \subseteq^D \), because in this case, for all \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X \sigma_{y_1}(\delta) \) 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 \( q_1 \subseteq^D q \).

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

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 \( \preceq \). Moreover, as shown in Theorem 4.1 below, \( \preceq \) is optimal in this respect.

**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 \( \text{inst}_{FD}(U) \), \( \sup_{\Delta}(q_1) \leq \sup_{\Delta}(q) \).

**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 3.1(1), because in this case, \( X \rightarrow X_1 \in FD^+ \).
- If \( y_1 \neq \top \), then, the case where \( y_1 \not\in \pi_{Y_1} (\Delta) \) is trivial because we have \( sup_\Delta (q_1) = 0 \). So, let us assume that \( y_1 \in \pi_{Y_1} (\Delta) \). 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 3.1(1), we obtain that \( |\pi_X \sigma_{y_1} (\Delta)| \leq |\pi_X \sigma_y (\Delta)| \). On the other hand, if \( y \) is a subtuple of \( y_1 \), \( |\pi_X \sigma_y (\Delta)| \leq |\pi_X \sigma_y (\Delta)| \), and thus, we obtain \( ans_\Delta (q_1) \leq ans_\Delta (q) \). 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 \( inst_{FD} (U) \) 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 \( Y \neq \emptyset \) 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 \( y.A \neq y_1.A \). Let \( \Delta \) be a table over \( U \) containing a single tuple \( t \) such that \( t.Y_1 = y_1 \) and \( t.A \neq y_1.A \). Then, clearly, \( \Delta \) is in \( inst_{FD} (U) \) (because a table containing a single tuple satisfies any set of functional dependencies), and we have \( sup_\Delta (q_1) = 1 \) (because \( ans_\Delta (q_1) = \{ t.X_1 \} \) and \( sup_\Delta (q) = 0 \) (because, as \( t.Y \neq y \), \( ans_\Delta (q) = \emptyset \)). 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: \( t_1.\pi_{X_1} (Y_1)^+ = t_2.\pi_{X_1} (Y_1)^+ \), \( t_1.Y_1 = t_2.Y_1 = y_1 \), and, for every attribute \( A \) not in \( \pi_{X_1} (Y_1)^+ \) (note that such an attribute exists because, assuming that \( \pi_{X_1} (Y_1)^+ = U \) entails \( XY_1 \rightarrow X_1 \in FD^+ \)), \( t_1.A \neq t_2.A \). Then, as \( match(t_1, t_2) = \pi_{X_1} (Y_1)^+ \), 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 \( X_1 \not\subseteq \pi_{X_1} (Y_1)^+ \). Indeed, if \( X_1 \subseteq \pi_{X_1} (Y_1)^+ \), 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.

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 \( inst_{FD} (U) \).

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 \( inst_{FD} (U) \) is considered, which is the case in practice.

For instance, we recall from Example 1.1 that, in the given table \( \Delta \), we have \( sup_\Delta (\pi_U \sigma_{p_2} (\delta)) \leq sup_\Delta (\pi_U \sigma_{beer} (\delta)) \). However, these two queries are not comparable according to \( \preceq \) because none of the tuples \( p_2 \) and \( beer \) 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 to all queries in \( Q (U) \) of the one given in [14]).

**Definition 4.2.** Let \( \Delta \) be in \( inst_{FD} (U) \) and \( q = \pi_X \sigma_y (\delta) \) and \( q_1 = \pi_X \sigma_y (\delta) \) 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 $\pi_{Y\gamma}(\Delta)$.

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 [14] 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:

- $\pi_{\text{Cid}} \sigma_{\text{Paris}}(\delta) \preceq_\Delta \pi_{\text{Cid}} \sigma_{\text{Paris beer}}(\delta)$, because $\text{Cid Caddr Ptype} \rightarrow \text{Cid}$ and $\text{Caddr Ptype} \rightarrow \text{Caddr}$ are in $FD^+$, and $\Delta_{\text{Caddr}}(\text{Paris beer}) = \text{Paris}$.

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

- $\pi_{\text{Cid}} \sigma_{\top}(\delta) \preceq_\Delta \pi_{\text{Cid}} \sigma_{\text{Paris}}(\delta)$, because $\text{Cid Caddr} \rightarrow \text{Cid}$ and $\text{Caddr} \rightarrow \emptyset$ are in $FD^+$.

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

- $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta) \preceq_\Delta \pi_{\text{Qty}} \sigma_{\text{p}_2}(\delta)$, because $\text{Cid Pid} \rightarrow \text{Qty}$ and $\text{Pid} \rightarrow \text{Ptype}$ are in $FD^+$, and $\Delta_{\text{Ptype}}(\text{p}_2) = \text{beer}$.

In this case however, $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta)$ and $\pi_{\text{Qty}} \sigma_{\text{p}_2}(\delta)$ are not comparable according to $\preceq$.

- $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta) \preceq_\Delta \pi_{\text{Qty}} \sigma_{\text{p}_2 \text{milk}}(\delta)$, because the fact that $\text{p}_2 \text{milk}$ is not in $\pi_{\text{Pid} \text{Ptype}}(\Delta)$ entails that $\pi_{\text{Qty}} \sigma_{\text{p}_2 \text{milk}}(\delta) \in Q(U) \setminus Q(\Delta)$.

Again in this case, $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta)$ and $\pi_{\text{Qty}} \sigma_{\text{p}_2 \text{milk}}(\delta)$ are not comparable according to $\preceq$.

- $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta) \preceq_\Delta \pi_{\text{Ptype}} \sigma_{\text{c}_1 \text{p}_2}(\delta)$ because $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta)$ and $\pi_{\text{Ptype}} \sigma_{\text{c}_1 \text{p}_2}(\delta)$ are in $Q(\Delta)$ and $\text{Cid Pid} \rightarrow \text{Ptype}$ is in $FD^+$.

As above, $\pi_{\text{Cid}} \sigma_{\text{beer}}(\delta)$ and $\pi_{\text{Ptype}} \sigma_{\text{c}_1 \text{p}_2}(\delta)$ are not comparable according to $\preceq$.

Given a table $\Delta$ in $\text{inst}_{FD}(U)$, based on Definition 4.2, it can be checked that:

- All queries not in $Q(\Delta)$ are the most specific queries with respect to $\Delta$ among all queries in $Q(U)$.
- For every possibly empty schema $X$ and every tuple $y$, $\pi_X \sigma_{\top}(\delta) \preceq_\Delta \pi_X \sigma_y(\delta)$.
- Given $q = \pi_X \sigma_y(\delta)$ and $q_1 = \pi_X \sigma_{y_1}(\delta)$ in $Q(\Delta)$ such that $X_1 \subseteq X$, $Y \subseteq Y_1$ and $y = y_1 Y$, $q \preceq_\Delta q_1$ holds for any set $FD$ of functional dependencies.

The following proposition shows that $\preceq_\Delta$ as defined above is a pre-ordering that refines $\preceq$. 

Proposition 4.2. The relation $\preceq_\Delta$ is a pre-ordering over $Q(U)$ that generalizes $\preceq$, that is, for all queries $q$ and $q_1$ in $Q(U)$, if $q \preceq q_1$ then, for every $\Delta$ in $\text{inst}_{FD}(U)$, $q \preceq_\Delta q_1$.

Proof: It is easy to see that $\preceq_\Delta$ is reflexive, thus we show the transitivity of $\preceq_\Delta$. 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 $q \preceq_\Delta q_1$ and $q_1 \preceq_\Delta q_2$. We first note that if $q_2$ is not in $Q(\Delta)$, then we have $q \preceq_\Delta q_2$. So, let us assume that $q_2$ is in $Q(\Delta)$. Then, as werough $q_1 \preceq_\Delta q_2$, we must have that $q_1$ is also in $Q(\Delta)$, and thus that $q$ is in $Q(\Delta)$ as well.

If $Y_2 \rightarrow X_2$ is in $FD^+$, then we have $q \preceq_\Delta q_2$. If $Y_2 \rightarrow X_2 \notin FD^+$, then we show that $Y_1 \rightarrow X_1$ cannot be in $FD^+$ either. Indeed, if $Y_1 \rightarrow X_1 \in FD^+$ then, as we assume that $q_1 \preceq_\Delta q_2$, $X_1Y_2 \rightarrow X_2$ and $Y_2 \rightarrow Y_1$ are in $FD^+$. So, $Y_2 \rightarrow X_1$ is in $FD^+$, and thus, $Y_2 \rightarrow X_2 \in FD^+$, which is a contradiction. Thus, in this case, we also have that $Y \rightarrow X$ is not in $FD^+$.

Therefore, the following dependencies are in $FD^+$: $XY_1 \rightarrow X_1, Y_1 \rightarrow Y, X_1Y_2 \rightarrow X_2$ and $Y_2 \rightarrow Y_1$. As a consequence, $XY_2 \rightarrow XY_1 \in FD^+$, and so, using $XY_1 \rightarrow X_1$, $XY_2 \rightarrow X_1Y_2$ is in $FD^+$. As $X_1Y_2 \rightarrow X_2 \in FD^+$, we obtain that $XY_2 \rightarrow X_2 \in FD^+$. On the other hand, as $Y_2 \rightarrow Y_1$ and $Y_1 \rightarrow Y$ are in $FD^+$, so is $Y_2 \rightarrow Y$. Moreover, as we have $y = \Delta_Y(y_1)$ and $y_1 = \Delta_Y(x_2)$, we obtain $y = \Delta_Y(y_2)$, which shows that $q \not\preceq_\Delta q_2$.

Let us now assume that $q$ and $q_1$ are such that $q \preceq q_1$. If $q_1$ is not in $Q(\Delta)$, then we have $q \preceq_\Delta q_1$. Assuming now that $q_1$ is in $Q(\Delta)$, since $y$ is a subtuple of $y_1$, $q$ is in $Q(\Delta)$ as well. Moreover, in this case, $Y \subseteq Y_1$, and so, $Y_1 \rightarrow Y \in FD^+$ and $\Delta_Y(y_1) = y$ (as $y$ is a subtuple of $y_1$). Since $q \preceq q_1$ also implies that $XY_1 \rightarrow X_1$ be in $FD^+$, we have $q \preceq_\Delta q_1$, and thus, the proof is complete.

As was the case for the pre-ordering $\preceq$, it is easy to see that $\preceq_\Delta$ is not an ordering. This is so because the two distinct queries $q = \pi_{C_id}\sigma_\top(\delta)$ and $q_1 = \pi_{C_id,C_addr}\sigma_\top(\delta)$ of Example 4.1 are such that $q \preceq q_1$ and $q_1 \preceq q$, and thus, by Proposition 4.2, $q \preceq_\Delta q_1$ and $q_1 \preceq_\Delta q$.

The following example shows how the pre-ordering $\preceq_\Delta$ behaves in the simple case of Example 4.2.

Example 4.4. As in Example 4.2, let $U = \{A, B\}$ and $FD = \emptyset$. Assume that $\text{dom}(A) = \{a\}$ and $\text{dom}(B) = \{b\}$ and the database to be considered is $\Delta = \{ab\}$. Then, the sets $Q(U)$ and $Q(\Delta)$ are equal and contain all queries $\pi_X\sigma_y(\delta)$ where $X \in \{\emptyset, A, B, AB\}$ and $y \in \{\top, a, b, ab\}$.

In this case, the pre-ordering $\preceq_\Delta$ 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 $q \preceq_\Delta q_1$ and where two queries $q$ and $q_1$ in the same box are such that both $q \preceq_\Delta q_1$ and $q_1 \preceq_\Delta q$ hold.

The graph above clearly shows more comparisons than the graph of Example 4.2, which results in a simpler structure. We shall come back to this consequence of Proposition 4.2 later in the paper.
As a consequence of Proposition 4.2, it is easy to see \( \preceq_{\Delta} \) generalizes query containment ([19]) and diagonal containment ([11]), because we have seen that so does \( \preceq \). Moreover, all comparison cases mentioned after Definition 4.1 also hold in this case. That is, for all possibly empty schemas \( X \) and \( X_1 \) and every tuple \( y \), we have:

- If \( y \) is a subtuple of \( y_1, \pi_X \sigma_y(y) \preceq_{\Delta} \pi_X \sigma_y(y_1) \), and in particular \( \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(y_1)(\delta) \). Moreover, if \( y \not\in \pi_X(\Delta) \) (which entails that \( y_1 \not\in \pi_X(\Delta) \)) we also have \( \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(y) \).

- If \( X \rightarrow X_1 \) is in \( FD^+ \) then \( \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(y)(\delta) \), and in particular, \( \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(\delta) \). Moreover, if \( y \not\in \pi_X(\Delta) \) we also have \( \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(y) \).

- If \( Y \rightarrow X \) is in \( FD^+ \), \( \pi_0 \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(\delta) \). In this case, we also have \( \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(\delta) \) and \( \pi_0 \sigma_y(y) \preceq_{\Delta} \pi_X \sigma_y(y) \).

- If \( Y \rightarrow X \) is in \( FD^+ \), \( \pi_0 \sigma_y(y) \preceq_{\Delta} \pi_X \sigma_y(y) \), meaning that, as is the case for \( \preceq \), \( \pi_U \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_y(y) \).

- If \( u \) is a tuple over \( U \) and \( y \) is a subtuple of \( u, \pi_X \sigma_y(\delta) \preceq_{\Delta} \pi_X \sigma_u(y) \). Moreover, if \( y \not\in \pi_X(\Delta) \) (which entails that \( u \not\in \pi_X(\Delta) \)) we also have \( \pi_X \sigma_u(\delta) \preceq_{\Delta} \pi_X \sigma_y(y) \).

Of course, the fact that \( \preceq_{\Delta} \) 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 \( \preceq_{\Delta} \). This is precisely what is stated in the following proposition.

**Proposition 4.3.** Let \( \Delta \) be a table in \( \text{inst}_{FD}(U) \). For all queries \( q \) and \( q_1 \) in \( \mathcal{Q}(U) \), if \( q \preceq_{\Delta} q_1 \) then \( \sup_{\Delta}(q_1) \leq \sup_{\Delta}(q) \).

**Proof:**

Let \( q = \pi_X \sigma_y(\delta) \) and \( q_1 = \pi_X \sigma_y(y_1)(\delta) \) be two queries in \( \mathcal{Q}(U) \). We first note that the result is trivial if \( q_1 \not\in \mathcal{Q}(\Delta) \). So let us assume that \( q_1 \in \mathcal{Q}(\Delta) \), and thus that \( q \in \mathcal{Q}(\Delta) \). 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 \( q_1 \) and \( q \) are in \( \mathcal{Q}(\Delta) \) 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 3.1(1). If \( y_1 \neq \top \), the table \( \sigma_y(y_1) \) satisfies \( X \rightarrow Y_1 \), and as \( XY_1 \rightarrow X_1 \) is in \( FD^+ \), \( \sigma_y(y_1) \) satisfies \( X \rightarrow X_1 \). Thus, by Lemma 3.1(1), \( |\pi_X \sigma_y(y_1)(\Delta)| \leq |\pi_X \sigma_y(\delta)| \). As \( Y_1 \rightarrow Y \in FD^+ \) and \( y = \Delta_Y(y_1) \), by Lemma 3.1(2), we have \( |\pi_X \sigma_y(\Delta)| \leq |\pi_X \sigma_y(\delta)| \). 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

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

To see this, we recall from Example 1.1 that, when considering the table \( \Delta \) given in this example, we have \( \pi_U \sigma_{p_2}(\delta) \preceq_{\Delta} \pi_U \sigma_{\text{beer}}(\delta) \). On the other hand, in the table \( \Delta' \) obtained from \( \Delta \) by replacing all occurrences of beer by wine, we have \( \sup_{\Delta'}(\pi_U \sigma_{\text{beer}}(\delta)) < \sup_{\Delta'}(\pi_U \sigma_{p_2}(\delta)) \), although \( \Delta' \in \text{inst}_{FD}(U) \).
Nevertheless, $\preceq_\Delta$ can be shown to characterize the anti-monotonicity of the support, provided that we consider only tables in $\text{inst}_F(D)(U)$ in which $q$ and $q_1$ have the same ‘status’ as in $\Delta$. In other words, we show that

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

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 $\text{inst}_F(D)(U)$ and let $q = \pi_X\sigma_y(\delta)$ and $q_1 = \pi_{X_1}\sigma_{y_1}(\delta)$ be in $Q(U)$. The set of all tables in which $q$ and $q_1$ have the same ‘status’ as in $\Delta$, denoted by $\text{inst}_F(D)(\Delta, q, q_1)$, is the set of all tables $\Delta'$ in $\text{inst}_F(D)(U)$ such that:

- $q \in Q(\Delta')$ if and only if $q \in Q(\Delta)$,
- $q_1 \in Q(\Delta')$ if and only if $q_1 \in Q(\Delta)$, and
- if $q$ and $q_1$ are in $Q(\Delta)$ and $Y_1 \rightarrow Y \in FD^+$, then $\Delta_Y(y_1) = \Delta'_Y(y_1)$.

It is easy to see from Definition 4.3 that, for every $\Delta$ in $\text{inst}_F(D)(U)$ and all queries $q$ and $q_1$ in $Q(U)$, $\Delta$ is in $\text{inst}_F(D)(\Delta, q, q_1)$, and for every $\Delta'$ in $\text{inst}_F(D)(\Delta, q, q_1)$, $\text{inst}_F(D)(\Delta, q, q_1) = \text{inst}_F(D)(\Delta', q, q_1)$. Moreover, it should be clear from Definition 4.3 that testing whether $\Delta'$ belongs to $\text{inst}_F(D)(\Delta, q, q_1)$ 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 $\text{inst}_F(D)(U)$ and let $q$ and $q_1$ in $Q(U)$. Then, for every $\Delta'$ in $\text{inst}_F(D)(\Delta, q, q_1)$, $q \preceq_\Delta q_1$ 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 $\text{beer}$ by $\text{wine}$. Then, it should be clear that for $q = \pi_2\sigma_{p_2}(\delta)$ and $q_1 = \pi_2\sigma_{\text{beer}}(\delta)$, $\Delta'$ is not in $\text{inst}_F(D)(\Delta, q, q_1)$. However, if we consider $q' = \pi_1\sigma_{p_1}(\delta)$ and $q'_1 = \pi_1\sigma_{\text{milk}}(\delta)$, then $\Delta'$ does belong to $\text{inst}_F(D)(\Delta, q, q'_1)$.

The following theorem characterizes $\preceq_\Delta$ with respect to all tables in which the queries under comparison have the same ‘status’ as in $\Delta$.

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

**Proof:**

See Appendix A. \qed

Referring back to our running example, we have seen in Example 4.3 that $q = \pi_{C_1d}\sigma_{\text{beer}}(\delta)$ and $q_1 = \pi_{Q_1q_2}\sigma_{p_2}(\delta)$ 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 $\text{inst}_F(D)(\Delta, q, q_1)$, i.e., in every $\Delta'$ of $\text{inst}_F(D)(U)$ such that $q$ and $q_1$ are in $Q(\Delta')$ and product $p_2$ is of type $\text{beer}$.

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 $\text{inst}_F(D)(U)$, $\sup_{\Delta'}(q_1) \leq \sup_{\Delta'}(q)$. 
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 \( q = \pi_A \sigma_{c}(\delta) \) and \( q_1 = \pi_A \sigma_{b'}(\delta) \), then \( \Delta \) and \( \Delta' \) are in \( \text{inst}_{FD}(U) \), \( q \not\leq_{\Delta} q_1 \), and we have:

1. \( q \in Q(\Delta), q_1 \not\in Q(\Delta) \) and \( \sup_{\Delta}(q_1) < \sup_{\Delta}(q) \)
2. \( q \not\in Q(\Delta'), q_1 \in Q(\Delta') \) and \( \sup_{\Delta'}(q) < \sup_{\Delta'}(q_1) \).

This shows that the first two conditions in Definition 4.3 are necessary in order to make sure that we consider tables with respect to which \( q \) 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 \( \text{inst}_{FD}(U) \), \( q \not\leq_{\Delta} q_1 \), and:

1. \( q \) and \( q_1 \) are both in \( Q(\Delta) \) and \( Q(\Delta') \), but \( \Delta_C(b') \neq \Delta'_C(b') \)
2. \( \sup_{\Delta}(q_1) < \sup_{\Delta}(q) \) and \( \sup_{\Delta'}(q) < \sup_{\Delta'}(q_1) \).

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 \( \leq \) and \( \leq_{\Delta} \) 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:

1. \( q \) and \( q_1 \) are said to be equivalent, denoted by \( q \equiv q_1 \), if \( q \leq q_1 \) and \( q_1 \leq q \).
2. 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 \( q \leq_{\Delta} q_1 \) and \( q_1 \leq_{\Delta} q \).

- 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 paper, the equivalence class \( Q(U) \setminus Q(\Delta) \) will be denoted by \( C_{\Delta}^q \).

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, \( q \not\leq_{\Delta} q_1 \). Thus, it can be seen from the proof of Proposition 4.2 that this entails that \( Y \rightarrow X \) is in \( FD^+ \). In the remainder of the paper, 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 \( C^q_{\Delta} \).
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 \( \text{inst}_{FD}(U) \), \( \sup_{\Delta}(q) = \sup_{\Delta}(q_1) \).

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

We recall from Example 4.1 that, in the context of our running example, \( \pi_{\text{Cid}\sigma_{\top}}(\delta) \preceq \pi_{\text{CidCadd}\sigma_{\top}}(\delta) \) and \( \pi_{\text{CidCadd}\sigma_{\top}}(\delta) \preceq \pi_{\text{Cid}\sigma_{\top}}(\delta) \) 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 \( \pi_{\text{Cid}\sigma_{\top}}(\delta) \preceq \pi_{\text{CidCname\ Pype}\sigma_{\top}}(\delta) \) and \( \pi_{\text{CidCname\ Pype}\sigma_{\top}}(\delta) \preceq \pi_{\text{Cid\ Cadd\ Pype}\sigma_{\top}}(\delta) \). 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 \( \text{inst}_{FD}(\Delta, \pi_{\text{Cid}\sigma_{\top}}(\delta), \pi_{\text{CidCname\ Pype}\sigma_{\top}}(\delta)) \).

Based on Corollary 5.1, given a class \( [q] \) in \( \mathcal{C}(U) \) (respectively \( [q]_{\Delta} \) in \( \mathcal{C}_{\Delta}(U) \)), we denote by \( \sup_{\Delta}([q]) \) (respectively \( \sup_{\Delta}([q]_{\Delta}) \)) 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 \( \mathcal{C}(\Delta) \) (respectively \( [q]_{\Delta} \) in \( \mathcal{C}_{\Delta}(U) \)), \( [q] \) (respectively of \( [q]_{\Delta} \)) is said to be \( \text{frequent} \) if \( \sup_{\Delta}([q]) \geq \text{min-sup} \) (respectively \( \sup_{\Delta}([q]_{\Delta}) \geq \text{min-sup} \)).

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

The following theorem states that the support of equivalence classes is anti-monotonic with respect to the partial orderings \( \preceq \) and \( \preceq_{\Delta} \) 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 \( \text{inst}_{FD}(U) \), \( \sup_{\Delta}([q_1]) \leq \sup_{\Delta}([q]) \).
2. If $\Delta$ is in $\text{inst}_{FD}(U)$, $[q]_\Delta \preceq_\Delta [q_1]_\Delta$ holds if and only if, for every $\Delta'$ in $\text{inst}_{FD}(\Delta, q, q_1)$, $\sup_{\Delta'}([q_1]_\Delta) \leq \sup_{\Delta'}([q]_\Delta)$.

**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 $[q_1]_\Delta = [q_1]_{\Delta'}$. Thus, we have $\sup_{\Delta'}([q]_\Delta) = \sup_{\Delta'}([q]_{\Delta'})$ and $\sup_{\Delta'}([q_1]_\Delta) = \sup_{\Delta'}([q_1]_{\Delta'})$, 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 $C(U)$ or $C_\Delta(U)$.

2. Frequent classes can be computed using the Apriori trick ([1]). In other words, as for individual queries, given a support threshold $\text{minsup}$, if $\sup_{\Delta'}([q]) < \text{min-sup}$ (respectively $\sup_{\Delta'}([q]_\Delta) < \text{min-sup}$), then for every class $[q_1]$ such that $[q] \preceq [q_1]$ (respectively $[q_1]_\Delta$ such that $[q]_\Delta \preceq [q_1]_\Delta$), we have that $\sup_{\Delta'}([q_1]) < \text{min-sup}$ (respectively $\sup_{\Delta'}([q_1]_\Delta) < \text{min-sup}$). Consequently, the support of $[q_1]$ (respectively $[q_1]_\Delta$) has not to be computed.

However, computing frequent classes instead of individual frequent queries is effective 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, which we define as in [19]. Formally, given a schema $X$, $K$ is said to be a key of $X$ if $K$ is a minimal (with respect to set inclusion) subset of $X$ such that $K \rightarrow X$ is in $FD^+$. The set of all keys of $X$ is denoted by $\text{keys}(X)$.

The following proposition characterizes the content of classes in $C(U)$.

**Proposition 5.1.** For every $q = \pi_X\sigma_y(\delta)$ in $Q(U)$, $[q]$ is the set of all queries $q_1 = \pi_X\sigma_{y_1}(\delta)$ 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 $q_1 = \pi_X\sigma_{y_1}(\delta)$ in $[q]$. It follows from Definition 4.1 that $(XY)^+ = (X_1Y_1)^+$ and $y_1 = y$. Thus, $Y_1 = Y$, and it can be seen that, as $(XY)^+ = (X_1Y)^+$, there exists $K \in \text{keys}(XY)$ such that $K \subseteq X_1Y \subseteq (XY)^+$. Therefore, $(K \setminus Y^+) \subseteq K \subseteq X_1Y$, 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, $\pi_{(K \setminus Y^+)}\sigma_y(\delta) \in Q$.

Conversely, for every $X_1$ such that $(K \setminus Y^+) \subseteq X_1 \subseteq (XY)^+$, $(X_1Y)^+ = (XY)^+$ because, as $K$ is in $\text{keys}(XY)$, we have $((K \setminus Y^+)^+) = (XY)^+$. Indeed, as $K = (K \setminus Y^+)^+Y^+ \subseteq ((K \setminus Y^+)^+)Y^+$, we have $((XY)^+) \subseteq ((K \setminus Y^+)^+)Y^+$. Moreover, as $K \subseteq XY$, $(K \setminus Y^+) \subseteq (XY)^+$, and so, $(K \setminus Y^+)Y \subseteq (XY \setminus Y^+)Y$. Since $(XY \setminus Y^+)Y \subseteq XY$, we obtain that $((K \setminus Y^+)^+) \subseteq (XY)^+$. Therefore, $(X_1Y)^+ = (XY)^+$ and thus, $\pi_X\sigma_y(\delta) \in [q]$, which completes the proof. \qed
To illustrate Proposition 5.1, let us consider \( q = \pi_{Cid,Ptype}\sigma_{p_2}(\delta) \) in the context of our running example. In this case, we have \( Y^+ = Pid\ Ptype \) and \((Cid\ Ptype\ Pid)^+ = U\), and thus, \([q]\) is the set of all queries \( \pi_{X_1}\sigma_{p_2}(\delta) \) such that \( Cid \subseteq X_1 \subseteq U \).

As a consequence of Proposition 5.1 above, for every \( q \in Q(U) \), \([q]\) contains exactly one representative \( \pi_{X}\sigma_g(\delta) \) such that \( X^+ = X \) and \( Y \subseteq X \). Therefore, \( C(U) \) is isomorphic to the set of all queries \( q = \pi_{X}\sigma_g(\delta) \) such that \( X^+ = X \) and \( Y \subseteq X \).

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

**Definition 5.2.** For every \( q = \pi_{X}\sigma_g(\delta) \) in \( Q(\Delta) \), the set of keys of \( q \), denoted by \( Keys(q) \), is the set of all queries \( y_0 = \pi_{X_0}\sigma_{y_0}(\delta) \) in \( Q(\Delta) \) such that

- \( X_0 = K_0 \setminus Y^+ \), where \( K_0 \in keys(XY) \) and
- \( y_0 = \Delta Y_0(y) \), where \( Y_0 \in keys(Y) \).

It follows from Definition 5.2 that, for every \( q = \pi_{X}\sigma_g(\delta) \) in \( Q(\Delta) \) such that \( Y \rightarrow X \) is in \( FD^+ \), \( Keys(q) = \{\pi_{0}\sigma_{y_0}(\delta) | Y_0 \in keys(Y) \land y_0 = \Delta Y_0(y)\} \). Indeed, in this case, \( X^+ \subseteq Y^+ \) and thus, for every \( X_0 \in keys(XY) \), \( X_0 \subseteq (XY)^+ = Y^+ \). Hence, \( X_0 \setminus Y^+ = \emptyset \).

**Example 5.1.** In the context of Example 1.1, let \( q = \pi_{Cid\ Cname}\sigma_\tau(\delta) \). As \( keys(Cid\ Cname) = \{Cid\} \), we have \( Keys(q) = \{\pi_{Cid}\sigma_\tau(\delta)\} \).

For \( q' = \pi_{Cid\ Ptype\ Qty\ Pid}(\delta) \), we have \( keys(Cid\ Ptype\ Qty\ Pid) = \{Cid\ Pid\} \) and \( keys(Pid) = \{Pid\} \). Since \( Pid^+ = Pid\ Ptype \), we obtain \( Keys(q') = \{\pi_{Cid\ Ptype}(\delta)\} \).

Now, for \( q = \pi_{Ptype}\sigma_{p_2}(\delta) \), since \( Pid \rightarrow Ptype \) and \( keys(Pid) = \{Pid\} \), we have \( Keys(q) = \{\pi_{0}\sigma_{p_2}(\delta)\} \).

The following proposition characterizes equivalent classes modulo \( \equiv_\Delta \).

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

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

   \[ [q]_\Delta = \{\pi_{X_1}\sigma_{y_1}(\delta) | (\exists \pi_{X_0}\sigma_{y_0}(\delta) \in Keys(q))((X_0 \subseteq X_1 \subseteq (X_0 Y_0)^+ \land (Y_0 \subseteq Y_1 \subseteq Y_0^+) \land (y_1 = \Delta Y_1(y)))\} \].

**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 \( q_1 = \pi_{X_1}\sigma_{y_1}(\delta) \) be in \( [q]_\Delta \). It follows from Definition 4.2 that \((XY)^+ = (X_1 Y_1)^+\), \( Y^+ = Y_1^+ \) and \( y_1 = \Delta Y_1(y) \).

It can be seen that there exists \( K \in keys(XY) \) such that \( K \subseteq X_1 Y_1 \subseteq (XY)^+ \) and \( Y_0 \in keys(Y) \) such that \( Y_0 \subseteq Y_1 \subseteq Y^+ \) and \( y_0 = \Delta Y_0(y) \). Moreover, for \( X_0 = K \setminus Y^+ \), as \( X_0 \cap Y_0^+ = \emptyset \) and \( Y_1 \subseteq Y_0^+ \), \( X_0 \cap Y_1 = \emptyset \). Since \( X_0 \subseteq K \subseteq X_1 Y_1 \), we have \( X_0 \subseteq X_1 Y_1 \), 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, \( (X_0Y_0)^+ = (X_1Y_1)^+ \) and \( Y_0^+ = Y_1^+ \). As we also have \( y_1 = \Delta y_1(y_0) \), it can be seen that this entails that \( q_1 \equiv \pi X_0 \sigma_{y_0}(\delta) \), and thus that \( q_1 \in [q]_\Delta \) (because it is easy to see that \( q_0 \equiv_{\Delta} q \)).

Thus, the proof is complete.

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

For instance, consider again the queries \( q \) and \( q' \) of Example 5.1. We have:

\[
[q]_\Delta = \{ \pi C id \sigma_T(\delta), \pi C id C name \sigma_T(\delta), \pi C id C addr \sigma_T(\delta), \pi C id C name C addr \sigma_T(\delta) \}
\]

\[
[q']_\Delta = \{ \pi X \sigma_y(\delta) \ | \ (C id \subseteq X \subseteq U) \land (y = p_2 \lor y = p_2 \text{ beer}) \}.
\]

Thus the equivalence classes \([q]_\Delta \) and \([q']_\Delta \) are respectively represented by \( \pi C id C name C addr \sigma_T(\delta) \) and \( \pi C id C name C addr \text{ Pid } P type \text{ Qty } \sigma_{p_2 \text{ beer}}(\delta) \) (or more simply, by \( \sigma_{p_2 \text{ beer}}(\delta) \)).

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 simple case where \( U = \{A, B\} \) and \( FD = \emptyset \). Assuming that \( \text{dom}(A) = \{a\} \) and \( \text{dom}(B) = \{b\} \) and considering the particular representatives as defined above, the set \( C(U) \) along with comparisons according to \( \preceq \) are shown below (case (a)).

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

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

We draw the attention that, even in this simple case without functional dependencies, the structures of \( C(U) \) and \( C_\Delta(U) \) are simpler than those of \( Q(U) \) and \( Q_\Delta(U) \). However, we point out that, although the previous two graphs show that, in this case, \( C(U) \) and \( C_\Delta(U) \) are lattices, this does not hold in general. This issue will be addressed shortly in the next section.

To end the section, we note that, since \( \preceq_\Delta \) refines \( \preceq \), given a table \( \Delta \) in \( \text{inst}_{FD}(U) \), equivalence classes modulo \( \equiv \) are smaller than those modulo \( \equiv_{\Delta} \). More formally, for every query \( q \) in \( Q(U) \), we have \( [q] \subseteq [q]_{\Delta} \).
This implies that, given \( \Delta \) in \( \text{inst}_{FD}(U) \), it is preferable to compute frequent classes modulo \( \equiv_{\Delta} \), because the cardinality of \( C(U) \) is less than that of \( C_{\Delta}(U) \). This is precisely what has been considered in our previous work [14].

6. On the Structures of \( C(U) \) and \( C_{\Delta}(U) \)

In this section, we first give an example showing that neither \( C(U) \) nor \( C_{\Delta}(U) \) has a lattice structure, in general. Then we identify particular cases in which the considered sets of equivalence classes have a lattice structure.

The counter example mentioned above, borrowed from [14], is the following.

Example 6.1. 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 \( q_1 = \pi_B \sigma_\top(\delta) \) 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 \( q = \pi_{BC} \sigma_\top(\delta) \) and \( q' = \pi_{BD} \sigma_\top(\delta) \), for \( i = 1, 2, [q] \preceq [q_i] \) and \( [q'] \preceq [q_i] \), i.e.,

\[
\begin{align*}
\pi_{BC} \sigma_\top(\delta) \preceq & \pi_B \sigma_\top(\delta) \quad \text{and} \quad \pi_{BC} \sigma_\top(\delta) \preceq \pi_{ABD} \sigma_a(\delta), \\
\pi_{BD} \sigma_\top(\delta) \preceq & \pi_B \sigma_\top(\delta) \quad \text{and} \quad \pi_{BD} \sigma_\top(\delta) \preceq \pi_{ABD} \sigma_a(\delta).
\end{align*}
\]

(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 \( X_0 = X_0^+, \ Y_0 = Y_0^+ \) and \( Y_0 \subseteq X_0 \) be such that, for \( i = 1, 2, [q] \preceq [q_i] \preceq [q_0] \preceq [q_i] \).

Then, we have \( [q] \neq [q_0] \) and:

1. \( X_0 \subseteq (BCY_0)^+, B \subseteq X_0 \) and \( ABD \subseteq X_0 \);

2. \( \top \) is a subtuple of \( y_0 \) and \( y_0 \) is a subtuple of \( \top \) and \( a \).

Then, by item 2 above, we obtain \( y_0 = \top \). So, taking into account that \( [q] \neq [q_0] \), thus that \( X_0 \neq BC \), item 1 above can be written as \( ABD \subseteq X_0 \subset BC \), which is 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 \( [q_i] \), we conclude that \( C(U) \) is not a lattice in this case.

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

\[
\begin{align*}
\pi_{BC} \sigma_\top(\delta) \preceq & \pi_B \sigma_\top(\delta) \quad \text{and} \quad \pi_{BC} \sigma_\top(\delta) \preceq \pi_{ABD} \sigma_a(\delta), \\
\pi_{BD} \sigma_\top(\delta) \preceq & \pi_B \sigma_\top(\delta) \quad \text{and} \quad \pi_{BD} \sigma_\top(\delta) \preceq \pi_{ABD} \sigma_a(\delta).
\end{align*}
\]

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 \( i = 1, 2, [q]_\Delta \preceq [q_0]_\Delta \preceq [q_i]_\Delta \) then we have \( [q]_\Delta \neq [q_0]_\Delta \) and:

1. \( X_0 \subseteq (BCY_0)^+, B \subseteq X_0 \) and \( ABD \subseteq X_0 \);

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 \( y_0 \) does not exist. As a consequence, the set \( C_{\Delta}(U) \) is not a lattice.

Despite this negative result, in [14], we have proposed algorithms for mining all frequent queries in the case of star schemas. We recall in this respect that star schemas are those according to which data warehouses are usually designed ([13, 16]). Thus, this case is important when mining queries from data warehouses.

We now focus on particular cases in which equivalence classes to be considered have a lattice structure: first we identify subsets of \( C(U) \) and \( C_{\Delta}(U) \) having this property, and second we show that, restricting the form of the dependencies in \( FD \) also results in the fact that \( C(U) \) and \( C_{\Delta}(U) \) have a lattice structure.

In order to prove the corresponding propositions, we define two operators over tuples, denoted by \( \cap \) and \( \sqcup \). This requires to introduce an additional tuple, which we call the contradictory tuple and denote by \( \bot \), and which is meant to lead to selection conditions that can not be satisfied by any tuple. This arises when, for instance, we consider a tuple that should have two distinct values over the same attribute. As for \( \top \), the schema of \( \bot \) is set as being the empty schema.

**Definition 6.1.** Let \( y_1 \) and \( y_2 \) be two tuples defined over \( Y_1 \) and \( Y_2 \), respectively.

1. \( y_1 \cap y_2 \) is the tuple over \( \text{match}(y_1, y_2) \) such that, for every \( A \) in \( \text{match}(y_1, y_2) \), \( (y_1 \cap y_2).A = y_1.A = y_2.A \).

2. \( y_1 \sqcup y_2 \) is the tuple defined by:
   - If \( \text{match}(y_1, y_2) = Y_1 \cap Y_2 \) then \( y_1 \sqcup y_2 \) is the tuple over \( Y_1Y_2 \) such that
     - for every \( A \) in \( Y_1 \), \( (y_1 \sqcup y_2).A = y_1.A \), and
     - for every \( A \) in \( Y_2 \), \( (y_1 \sqcup y_2).A = y_2.A \).
   - Otherwise, \( y_1 \sqcup y_2 = \bot \).

Intuitively, given two tuples \( y_1 \) and \( y_2 \), \( y_1 \cap y_2 \) is the ‘largest’ subtuple of \( y_1 \) and \( y_2 \) and \( y_1 \sqcup y_2 \), when different than \( \bot \), is the ‘smallest’ supertuple of \( y_1 \) and \( y_2 \). The following example illustrates Definition 6.1.

**Example 6.2.** Referring back to Example 1.1, for \( y_1 = (c_1 \ p_1 \ \text{Paris}) \) and \( y_2 = (p_1 \ \text{Paris} \ 1) \), we have \( y_1 \cap y_2 = (p_1 \ \text{Paris}) \) and \( y_1 \sqcup y_2 = (c_1 \ p_1 \ \text{Paris} \ 1) \).

On the other hand, for \( y_3 = (p_1 \ \text{Paris} \ 10) \), we have \( y_1 \cap y_3 = (\text{Paris}) \) and \( y_1 \sqcup y_2 = \bot \), because \( \text{match}(y_1, y_3) \neq Y_1 \cap Y_3 \).

As a last example, for \( y_4 = (\text{John} \ \text{milk}) \), we have \( \text{match}(y_1, y_4) = \emptyset \). Thus, \( y_1 \cap y_4 = \top \) and \( y_1 \sqcup y_4 = (c_1 \ p_1 \ \text{John} \ \text{Paris} \ \text{milk}) \).

Clearly, the introduction of the contradictory tuple \( \bot \) requires to consider all additional queries of the form \( \pi_X \sigma_{\bot}(\delta) \) whose answers in any \( \Delta \) of \( \text{inst}_{FD}(U) \) is assumed to be empty.

In order to extend the results in Theorem 4.1 and Theorem 4.2 accordingly, we assume that all these queries are more specific than any other query in \( Q(U) \) with respect the two pre-orderings \( \preceq \) and \( \preceq_{\Delta} \).

This implies that all these queries form a new equivalence class modulo \( \equiv \), which we denote by \( C_{\bot} \). Regarding \( \preceq_{\Delta} \), we consider all these queries as equivalent to all queries in \( C_{\bot}^{\Delta} \), and thus all these queries are assumed to be in the equivalence class \( C_{\bot}^{\Delta} \).
It should be noticed that the assumptions above are consistent with Corollary 5.1. Indeed, in the case of $\mathcal{C}(U)$, the queries of the form $\pi_X \sigma_Y(\delta)$ constitute the only equivalence class whose support is equal to 0 in every table of $\text{inst}_{FD}(U)$ (namely, the class $C_{\perp}$), and, in the case of $\mathcal{C}_\Delta(U)$, these queries are set to belong to the equivalence class containing all queries having a support equal to 0 (namely the class $C^0_{\Delta}$).

We are now ready to state the main results of this section. First, the proposition below states that all equivalence classes obtained with a fixed tuple in the selection form a lattice. We note in this respect that this result is a generalization of a similar result stated in [15], where no selections are considered.

**Proposition 6.1.** Let $y_0$ be a tuple. Then:

1. The set $\mathcal{C}(U, y_0) = \{ [\pi_X \sigma_Y(\delta) ] \mid X \subseteq U \}$ is a lattice with respect to $\preceq$.

2. If $\Delta \in \text{inst}_{FD}(U)$, the set $\mathcal{C}_\Delta(U, y_0) = \{ [\pi_X \sigma_Y(\delta)] \Delta \mid X \subseteq U \}$ is a lattice with respect to $\preceq_{\Delta}$.

**Proof:**

1. $\mathcal{C}(U, y_0)$ is the set of all classes of $\mathcal{C}(U)$ whose representatives $\pi_X \sigma_Y(\delta)$ are such that $X = X^+$ and $Y_0 \subseteq X$. Given $q_1 = \pi_{X_1} \sigma_{Y_0}$ and $q_2 = \pi_{X_2} \sigma_{y_0}$ in $\mathcal{C}(U, y_0)$, let $q^\uparrow = \pi_{X_1 \cap X_2} \sigma_{y_0}(\delta)$ and $q^\downarrow = \pi_{(X_1 \setminus X_2)} \sigma_{y_0}(\delta)$. Clearly, $q^\uparrow$ and $q^\downarrow$ are in $\mathcal{C}(U, y_0)$, and, for $i = 1, 2$, we have $q_i \preceq q^\uparrow$ and $q^\downarrow \preceq q_i$.

   Let $q = \pi_X \sigma_{y_0}(\delta)$ be in $\mathcal{C}(U, y_0)$ such that $q_i \preceq q$, for $i = 1, 2$. Then, for $i = 1, 2$, $X_i Y_0 \rightarrow X$ are in $FD^+$, that is, $X^+ \subseteq (X_i Y_0)^+$. Since $X = X^+$, $X_i = X_i^+$ and $Y_0 \subseteq X_i$, we obtain $X \subseteq X_i$, and so, $X \subseteq X_1 \cap X_2$. Therefore, $q \preceq q^\uparrow$, meaning that $q^\uparrow$ is the least upper bound of $q_1$ and $q_2$ in $\mathcal{C}(U, y_0)$.

   Now, let $q = \pi_X \sigma_{y_0}(\delta)$ be in $\mathcal{C}(U, y_0)$ such that $q \preceq q_i$, for $i = 1, 2$. Then, for $i = 1, 2$, $X Y_0 \rightarrow X_i$ are in $FD^+$, that is, $X_i^+ \subseteq (X Y_0)^+$. Since $X = X^+$, $X_i = X_i^+$ and $Y_0 \subseteq X_i$, we obtain $X_i \subseteq X$, and so, $(X_1 X_2)^+ \subseteq X$. Therefore, $q \preceq q^\downarrow$, meaning that $q^\downarrow$ is the greatest lower bound of $q_1$ and $q_2$ in $\mathcal{C}(U, y_0)$. Thus, $\mathcal{C}(U, y_0)$ is a lattice.

2. Regarding $\mathcal{C}_\Delta(U, y_0)$, let $\Delta$ be in $\text{inst}_{FD}(U)$. We first note that if $y_0 \notin \pi_{Y_0}(\Delta)$, then $\mathcal{C}_\Delta(U, y_0) = \{ C^0_\Delta \}$, which is trivially a lattice. Let us now consider the case where $y_0 \in \pi_{Y_0}(\Delta)$. We may assume that $Y_0 = Y_0^+$, in which case $\mathcal{C}_\Delta(U, y_0) \setminus \{ C^0_\Delta, C^1_\Delta \}$ is the set of all classes of $\mathcal{C}(U) \setminus \{ C^0_\Delta, C^1_\Delta \}$ whose representatives $\pi_X \sigma_{y_0}(\delta)$ are such that $X = X^+$, $Y_0 = Y_0^+$ and $Y_0 \subseteq X$. Given $q_1 = \pi_{X_1} \sigma_{y_0}$ and $q_2 = \pi_{X_2} \sigma_{y_0}$ in $\mathcal{C}_\Delta(U, y_0)$, let $q^\uparrow$ and $q^\downarrow$ as follows:

   (a) If $Y_0 \rightarrow X_1$ and $Y_0 \rightarrow X_2$ are both in $FD^+$ then $q^\uparrow = q^\downarrow = C^1_\Delta$.

   (b) If $Y_0 \rightarrow X_1 \in FD^+$ but $Y_0 \rightarrow X_2 \notin FD^+$ then $q^\uparrow = q_1$ and $q^\downarrow = q_2$.

   (c) If $Y_0 \rightarrow X_2 \in FD^+$ but $Y_0 \rightarrow X_1 \notin FD^+$ then $q^\uparrow = q_2$ and $q^\downarrow = q_1$.

   (d) Otherwise, $q^\uparrow = \pi_{X_1 \cap X_2} \sigma_{y_0}(\delta)$ and $q^\downarrow = \pi_{(X_1 \setminus X_2)} \sigma_{y_0}(\delta)$.

In case (a), we have $q_1 = q_2 = C^1_\Delta$, and so, $q^\uparrow$ and $q^\downarrow$ are trivially the least upper bound and the greatest lower bound of $q_1$ and $q_2$, respectively. In case (b), we have $q_2 \preceq q_1$, and, again, $q^\uparrow$ and $q^\downarrow$ are trivially the least upper bound and the greatest lower bound of $q_1$ and $q_2$, respectively. As case (c) is similar to case (b), we omit it. In case (d), the only difference between this case and the proof of (1) above is that $Y_0 = Y_0^+$. Thus, in this case, $q^\uparrow$ and $q^\downarrow$ are respectively the least upper bound and the greatest lower bound of $q_1$ and $q_2$, for the same reasons as in (1) above. Thus, the proof is complete. □

The next proposition shows that the set of equivalence classes has a lattice structure if we consider queries whose projections are restricted to hold on a fixed schema.
Proposition 6.2. Let $X_0$ be a schema. Then

1. The set $C(U, X_0)$ of all equivalence classes in $C(U)$ of the form $[\pi_{X_0}\sigma_y(\delta)]$ is a lattice with respect to $\preceq$.

2. If $\Delta \in \text{inst}_{FD}(U)$, the set $C_\Delta(U, X_0)$ of all equivalence classes in $C_\Delta(U)$ of the form $[\pi_{X_0}\sigma_y(\delta)]_\Delta$ is a lattice with respect to $\preceq_\Delta$.

Proof:

1. $C(U, X_0)$ is the set of all classes of $C(U)$ whose representatives $\pi_X\sigma_y(\delta)$ are such that $X = (X_0Y)^+$. Given $q_1 = \pi_{X_1}\sigma_{y_1}$ and $q_2 = \pi_{X_2}\sigma_{y_2}$ in $C(U, X_0)$, let $q^1$ be defined by:

   - If $y_1 \cup y_2 = \bot$, then $q^1 = C_{\bot}$
   - Else $q^1 = \pi_{(X_0y_1y_2)}\sigma_{(y_1\cup y_2)}(\delta)$.

Moreover, denoting by $Y_m$ the set $\text{match}(y_1, y_2)$, let $q^1 = \pi_{(X_0Y_m)}\sigma_{(y_1\cap y_2)}(\delta)$. Clearly, $q^1$ and $q^1$ are in $C(U, X_0)$, and, for $i = 1, 2$, we have $q_i \preceq q^1$ and $q^1 \preceq q_i$.

Let $q = \pi_X\sigma_y(\delta)$ be in $C(U, X_0)$ such that $q_i \preceq q$, for $i = 1, 2$. Then, for $i = 1, 2$, $y_i$ is a subtuple of $y$. So, if $y_1 \cup y_2 = \bot$, then $y$ is also equal to $\bot$ and thus, $q = q^1 = C_{\bot}$. If $y_1 \cup y_2 \neq \bot$, then we also have that, for $i = 1, 2$, $X_i \rightarrow X$ are in $FD^+$, that is, $X^+ \subseteq (X_iY)^+$. Since $X = (X_0Y)^+$, $X_i = (X_0Y_i)^+$ and $Y_i \subseteq Y$, we obtain $X \subseteq (X_0Y_1Y_2)^+$. Therefore, $q^1 \preceq q$, meaning that $q^1$ is the least upper bound of $q_1$ and $q_2$ in $C(U, X_0)$.

Now, let $q = \pi_X\sigma_y(\delta)$ be in $C(U, X_0)$ such that $q \preceq q_i$, for $i = 1, 2$. Then, $y$ is a subtuple of $y_i$ ($i = 1, 2$), and so $y$ is a subtuple of $y_1 \cap y_2$. Thus, $Y \subseteq Y_m$, and so, $(XY_m)^+ = (X_0Y_1Y_m)^+ = (X_0Y_m)^+$. As $X \subseteq (XY_m)^+ = (X_0Y_m)^+$, $X \rightarrow (X_0Y_m)^+$ is in $FD^+$, which entails that $q \preceq q^1$. Therefore, $q^1$ is the greatest lower bound of $q_1$ and $q_2$ in $C(U, X_0)$, and so, $C(U, X_0)$ is a lattice.

2. Regarding $C_\Delta(U, X_0)$, let $\Delta$ be in $\text{inst}_{FD}(U)$. Then, $C_\Delta(U, X_0) \setminus \{C_{\Delta_0}, C_{\Delta_1}\}$ is the set of all classes of $C_\Delta(U) \setminus \{C_{\Delta_0}, C_{\Delta_1}\}$ whose representatives $\pi_X\sigma_y(\delta)$ are such that $X = (X_0Y)^+, Y = Y_0^+$ and $Y \subseteq X$.

Given two comparable classes with respect to $\preceq_\Delta$ in $C_\Delta(U, X_0)$, $q_1 = \pi_{X_1}\sigma_{y_1}(\delta)$ and $q_2 = \pi_{X_2}\sigma_{y_2}(\delta)$, let $q^1$ and $q^1$ be defined as follows: If $q_1 \preceq_\Delta q_2$ (respectively $q_2 \preceq_\Delta q_1$) then $q^1 = q_2$ (respectively $q_1$) and $q^1 = q_1$ (respectively $q_2$). Then, in this case, we trivially have that $q^1$ and $q^1$ are the least upper bound and the greatest lower bound of $q_1$ and $q_2$, respectively.

Let us now assume that $q_1$ and $q_2$ are not comparable with respect to $\preceq_\Delta$. Then for $i = 1, 2$, $q_i \not\in \{C_{\Delta_0}, C_{\Delta_1}\}$, in which case, denoting by $Y_m$ the set $\text{match}(y_1, y_2)$, we define $q^1$ and $q^1$ as follows:

   - If $y_1 \cup y_2 = \bot$ then $q^1 = C_{\Delta}^0(U)$, else $q^1 = \pi_{(X_0y_1y_2)}\sigma_{(y_1\cup y_2)}(\delta)$.

Then, clearly, the only difference between this case and the proof of (1) above is that, for $i = 1, 2$, $Y_i = Y_i^+$. Thus, in this case $q^1$ and $q^1$ are respectively the least upper bound and the greatest lower bound of $q_1$ and $q_2$ for the same reasons as in (1) above. Thus, the proof is complete.

The next proposition states that if the set $FD$ is such that the closure is stable under union (i.e., if for all $X$ and $Y$, $(XY)^+ = X^+Y^+$), then the sets $C(U)$ and $C_\Delta(U)$ are lattices.

Proposition 6.3. If for all schemas $X$ and $Y$, we have $(XY)^+ = X^+Y^+$, then

1. $C(U)$ is a lattice with respect to $\preceq$, and

2. $C_\Delta(U)$ is a lattice with respect to $\preceq_\Delta$. 

Proof:
1. Let us first prove that, under the restriction stated in the proposition, \( C(U) \) is a lattice. Given \( q_1 = \pi_X \sigma_{y_1} \) and \( q_2 = \pi_X \sigma_{y_2} \) in \( C(U) \), let \( q^\uparrow \) be defined by:
   
   - If \( y_1 \sqcup y_2 = \bot \), then \( q^\uparrow = C_\bot \)
   - Else \( q^\uparrow = \pi_{((X_1 \cap X_2) Y_1^+ Y_2^+) \sigma_{(y_1 \sqcup y_2)}}(\delta) \).

   Then, as for \( i = 1, 2 \), \( X_i = X_i^+ \), we have \( ((X_1 \cap X_2) Y_1^+ Y_2^+) = ((X_1^+ \cap X_2^+) Y_1^+ Y_2^+) \), and so, using the restriction on functional dependencies and the fact that \( (X_1^+ \cap X_2^+) = (X_1 \cap X_2)^+, ((X_1 \cap X_2) Y_1^+ Y_2^+) = ((X_1 \cap X_2) (Y_1^+ Y_2^+))^+ \). Since we also have that \( Y_1 Y_2 \subseteq ((X_1 \cap X_2) Y_1^+ Y_2^+) \), \( q^\uparrow \) is one of the chosen representatives of a class in \( C(U) \). Moreover, for \( i = 1, 2 \), we have \( q_i \leq q^\uparrow \) (because, for \( i = 1, 2 \), \( y_i \) is a subtuple of \( y_1 \sqcup y_2 \) and \( ((X_1 \cap X_2) Y_1^+ Y_2^+) \subseteq X_i Y_1^+ Y_2^+ \).

   Now, denoting by \( Y_m \) the set \( \text{match}(y_1, y_2) \), let \( q^\downarrow = \pi_{X_{12} \sigma_{(y_1 \cap y_2)}}(\delta) \), where \( X_{12} = (X_1 \setminus Y_1^+ Y_1^+) X_2 Y_2^+ \). Thanks to the restriction on the functional dependencies, \( X_{12} = X_{12} \) and thus, \( q^\downarrow \) is one of the chosen representatives of a class in \( C(U) \). Moreover, for \( i = 1, 2 \), \( y_1 \cap y_2 \) is a subtuple of \( y_i \), and \( X^+_i = X_i \subseteq X_{12} Y_i^+ \). Thus, for \( i = 1, 2 \), \( q^\downarrow \leq q_i \).

   Let \( q = \pi_X \sigma_y(\delta) \) be in \( C(U) \) such that \( q_i \leq q \), for \( i = 1, 2 \). Then, for \( i = 1, 2 \), \( y_i \) is a subtuple of \( y \). So, if \( y_1 \sqcup y_2 = \bot \), then \( y \) is also equal to \( \bot \) and thus, \( q = q^\uparrow = C_\bot \). If \( y_1 \sqcup y_2 \neq \bot \), then \( y_1 \cap y_2 \) is a subtuple of \( y \). Moreover, for \( i = 1, 2 \), we also have that \( X_i Y \rightarrow X \) are in \( FD^+ \), that is, \( X^+ \subseteq (X Y)^+ \). Since \( X = X^+ \), \( X_i = X_i^+ \) and \( (X Y)^+ = X^+ Y^+ \), we obtain \( X \subseteq X_i Y^+ \), which implies that \( X \subseteq (X_1 \cap X_2) Y^+ \). Consequently, \( (X_1 \cap X_2) Y \rightarrow X \) is in \( FD^+ \), and so, \( q^\downarrow \leq q \). Thus, \( q^\downarrow \) is the least upper bound of \( q_1 \) and \( q_2 \) in \( C(U) \).

   Now, let \( q = \pi_X \sigma_y(\delta) \) be in \( C(U) \) such that \( q \leq q_i \), for \( i = 1, 2 \). Then, \( y \) is a subtuple of \( y_i \) (\( i = 1, 2 \)), and so \( y \) is a subtuple of \( y_1 \cap y_2 \). As for \( i = 1, 2 \), \( X Y_i \rightarrow X_i \) is in \( FD^+ \), \( X_i^+ \subseteq (X Y_i)^+ \). Since for \( i = 1, 2 \), \( X_i^+ = X_i \), \( X^+ = X \) and \( (X Y_i)^+ = X^+ Y_i^+ \), we obtain \( X_i \subseteq X Y_i^+ \). Therefore, for \( i = 1, 2 \), \( (X_i \setminus Y_i^+) \subseteq X \), which implies that \( (X_i \setminus Y_i^+) \subseteq X \). Hence, \( X_{12} \subseteq X Y_m^+ \), meaning that \( X Y_m^+ \rightarrow X_{12} \) is in \( FD^+ \), and so, \( q \leq q_i \). Therefore, \( q^\downarrow \) is the greatest lower bound of \( q_1 \) and \( q_2 \) in \( C(U) \), which shows that \( C(U, X_0) \) is a lattice. Thus, the first part of the proof is complete.

2. Given \( \Delta \) in \( \text{inst}_{FD}(U) \), let us prove that, under the restriction stated in the proposition, \( C_{\Delta}(U) \) is a lattice. Given two comparable classes with respect to \( \preceq_{\Delta} \) in \( C_{\Delta}(U) \), \( q_1 = \pi_X \sigma_{y_1}(\delta) \) and \( q_2 = \pi_X \sigma_{y_2}(\delta) \), let \( q^\uparrow \) and \( q^\downarrow \) be defined as follows:
   
   - If \( q_1 \preceq_{\Delta} q_2 \) (respectively \( q_2 \preceq_{\Delta} q_1 \)) then \( q^\uparrow = q_2 \) (respectively \( q_1 \)) and \( q^\downarrow = q_1 \) (respectively \( q_2 \)). In this case, we trivially have that \( q^\downarrow \) and \( q^\uparrow \) are the least upper bound and the greatest lower bound of \( q_1 \) and \( q_2 \), respectively.
   - If \( q_1 \) and \( q_2 \) are not comparable with respect to \( \preceq_{\Delta} \), then for \( i = 1, 2 \), \( q_i \notin \{ C_{\Delta}, C_{\Delta}^0 \} \), in which case, denoting by \( Y_m \) the set \( \text{match}(y_1, y_2) \), we define \( q^\downarrow \) and \( q^\uparrow \) as follows:
     
     - If \( y_1 \sqcup y_2 = \bot \) then \( q^\uparrow = C_{\Delta}^0(U) \), else \( q^\uparrow = \pi_{((X_1 \cap X_2) Y_1 Y_2) \sigma_{y_1 \sqcup y_2}}(\delta) \).
     - \( q^\downarrow = \pi_{(X_1 \sigma_{y_1 \cap y_2}}(\delta) \), where \( X_{12} = (X_1 \setminus Y_1^{+} (X_2 \setminus Y_2)^{+} Y_m^+ \).

   Then, clearly, the only difference between this case and the proof of (1) above is that, for \( i = 1, 2 \), \( Y_i = Y_i^+ \). Thus, in this case \( q^\downarrow \) and \( q^\uparrow \) are respectively the least upper bound and the greatest lower bound of \( q_1 \) and \( q_2 \) for the same reasons as in (1) above. Thus, the proof of this part is complete. □

We note that the restriction on closures in Proposition 6.3 above is equivalent to the fact that \( FD \) can be reduced so as to contain only dependencies whose left-hand-sides contain a single attribute. In particular, this restriction is satisfied when no functional dependency is considered, i.e., when \( FD = \emptyset \). Thus, when \( FD = \emptyset \), the sets \( C(U) \) and \( C_{\Delta}(U) \) are lattices, as seen in Example 5.2.
However, this restriction is a very particular case, which is clearly not satisfied in the case of our running example (because $\text{PidCid} \rightarrow \text{Qty}$ is in $FD$). More generally, it is easy to see that this restriction is not satisfied in the case of star schemas with more than one dimension, simply because of the dependency that states that the set of all dimension keys constitute a key of the fact table.

To end the section, we mention that it is currently unknown to the authors whether another relevant condition on the functional dependencies in $FD$ could lead to a result similar to that of Proposition 6.3.

7. Conclusion and Further Work

In this paper, we have considered the problem of mining all projection-selection queries from 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 paper.

Regarding possible extenstions of the present approach, we plan to investigate 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. It should be noticed in this respect that for star schemas, joins are always performed along keys and foreign keys.
- 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 plan to investigate 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


A. Proof of Theorem 4.2

The proof of the theorem is based on the following two lemmas.

Lemma A.1. Let $X$, $Y$, $X_1$ and $Y_1$ be attribute sets such that

- $Y_1 \rightarrow X_1 \notin FD^+$ and
- $XY_1 \rightarrow X_1 \notin FD^+$ or $Y_1 \rightarrow Y \notin FD^+$.

Then, for all tuples $y$ and $y_1$ over $Y$ and $Y_1$, respectively, there exists $\Delta$ in $inst_{FD}(U)$ 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 $FD^+$, then $X_1 \neq \emptyset$. Denoting by $Y_m$ the set $\mathit{match}(y,y_1)$, we successively consider the following two cases: (1) $Y_m = Y \cap Y_1$ and (2) $Y_m \subset Y \cap Y_1$.

(1) In the case where $Y_m = Y \cap Y_1$, we study separately the cases where $Y_1 \rightarrow Y$ is or not in $FD^+$.
(1.a) If $Y_1 \rightarrow Y$ is in $FD^+$, then $XY_1 \rightarrow X_1$ is not in $FD^+$. In other words, we have $Y^+ \subseteq Y_1^+$ and $X_1^+ \not\subseteq (XY_1)^+$. In this case, we have $(XY_1)^+ = Y_1^+$, $(XY_1)^+ = U$ implies that $(XY_1)^+ = U$, and thus that $X_1^+ \subseteq (XY_1)^+$, which is not possible.

Let $\Delta = \{ t, t_1 \}$ where $t$ and $t_1$ are two tuples over $U$ such that $t.(XY_1)^+ = t_1.(XY_1)^+$, $t.Y_1 = t_1.Y_1 = y_1$, $t.Y = y$, and for every $A \not\subseteq (XY_1)^+$ (such attributes exist since $(XY_1)^+ \neq U$), $t.A = t_1.A$. Therefore, by Proposition 3.1, $\Delta$ satisfies $FD$, and we have $\sup_{\Delta}(q) = 1$. Moreover, as seen above, $X_1^+ \not\subseteq (XY_1)^+$, and thus, $t.X_1 \neq t_1.X_1$. Thus, $\sup_{\Delta}(q_1) = 2$.

(1.b) If $Y_1 \rightarrow Y$ is not in $FD^+$, then $Y^+ \not\subseteq Y_1^+$, and so, $Y_1^+ \neq U$.

Let $\Delta = \{ t, t_1 \}$ where $t$ and $t_1$ are tuples over $U$ such that $t.Y_1^+ = t_1.Y_1^+$, $t.Y_1 = t_1.Y_1 = y_1$, $t.Y = y$, and for every $A \not\subseteq Y_1^+$ (such attributes exist since $Y_1^+ \neq U$), $t.A \neq t_1.A$. Therefore, by Proposition 3.1, $\Delta$ satisfies $FD$.

Moreover, assuming that $t.Y = t_1.Y$ implies that $Y \subseteq Y_1^+$ and thus that $Y^+ \subseteq Y_1^+$. Since this is not possible, we have $t.Y \neq t_1.Y$, which entails $\sup_{\Delta}(q) = 1$. On the other hand, as we also have that $X_1^+ \not\subseteq Y_1^+$, $t.X_1 \neq t_1.X_1$. Thus, $\sup_{\Delta}(q_1) = 2$.

(2) Let us now assume that $Y_m \subset Y \cap Y_1$, and let $y'$ be the tuple over $Y$ such that:
- for every $A \in (Y \setminus Y_1)$, $y'.A = y.A$,
- for every $A \in Y_m$, $y'.A = y.A = y_1.A$, and
- for every $A \in ((Y \cap Y_1) \setminus Y_m)$, $y'.A = y.A$.

Clearly, we have $\mathit{match}(y',y_1) = Y \cap Y_1$ and $y' \neq y$. Thus, denoting by $q'$ the query $\pi_X\sigma_{y'}(\delta)$, the proof above shows that there exists $\Delta' = \{ t, t_1 \}$ in $\mathit{inst}_{FD}(U)$ such that $\sup_{\Delta'}(q_1) = 2$, $\sup_{\Delta'}(q') = 1$ and $\sup_{\Delta'}(q) = 0$ (the last equality holds because, as $y' \neq y$, $q' \not\subseteq Q(\Delta')$).

Now, let $t_2$ be a tuple over $U$ such that $t_2.Y = y$, $t_2.Y_1^+ = t.Y_1^+$ and for every $A \not\subseteq Y_1^+$ (such attributes exist since otherwise, we would have $Y_m = Y \cap Y_1$), $t_2.A \neq t.A$ and $t_2.A \neq t_1.A$. We first note that, for $t_2$ to be defined, we must have $t.(Y_m \cap Y) = y.(Y_m \cap Y)$. This is so because:
- for every $A \in (Y_m^+ \cap Y) \setminus Y_1$, we have that $A \in Y_m$, which entails that $y.A = y'.A = t.A$, and
- for every $A \in (Y_m^+ \cap Y) \setminus Y_1$, we have $y.A = y'.A$ and, since $t.Y = y'$, we obtain $y.A = t.A$.

We show that $\Delta = \Delta' \cup \{ t_2 \}$ satisfies $FD$ and that $\sup_{\Delta}(q) = 1$ and $\sup_{\Delta}(q_1) = 2$. Indeed, as $\Delta'$ satisfies $FD$, we have $\mathit{match}(t, t_1) = (\mathit{match}(t, t_1))^+$, and by definition of $t_2$, we also have $\mathit{match}(t_2, t) = Y_m^+$. Moreover, in $\Delta'$, we have $t.Y_1 = t_1.Y_1$, and thus, $t.Y_m = t_1.Y_m$. Since this entails that $t.Y_1^+ = t_1.Y_1^+$, $t.Y_m^+ = t_1.Y_m^+$, and so, by construction of $t_2$, $\mathit{match}(t_2, t_1) = Y^+$. Hence, by Proposition 3.1, $\Delta$ satisfies $FD$.

Regarding the supports of $q$ and $q_1$ in $\Delta$, as $t_2$ is the only tuple such that $t_2.Y = y$, we have $\sup_{\Delta}(q) = 1$. On the other hand, as $\Delta' \subseteq \Delta$, we have $\sup_{\Delta}(q_1) \leq \sup_{\Delta}(q_1)$, and thus $\sup_{\Delta}(q_1) \geq 2$.

Therefore, we obtain that $\sup_{\Delta}(q_1) > \sup_{\Delta}(q)$, and this completes the proof.

\[\begin{align*}
\textbf{Lemma A.2.} \text{ Let } & \Delta \text{ be in } \mathit{inst}_{FD}(U), \text{ and } q = \pi_X \sigma_y(\delta) \text{ and } q_1 = \pi_X \sigma_{y_1}(\delta) \text{ in } Q(\Delta) \text{ such that } Y_1 \rightarrow X_1 \not\in FD^+, Y_1 \rightarrow Y \in FD^+, \text{ and } y \neq Y_1(y_1). \text{ Then there exists } \Delta' \in \mathit{inst}_{FD}(\Delta, q, q_1) \text{ such that } \sup_{\Delta'}(q_1) > \sup_{\Delta'}(q). \end{align*}\]
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 \( Y_1 \to X_1 \in \text{FD}^+ \)) and \( Y^+_m \neq U \) (because \( Y^+_m = U \) entails that \( y = \Delta_Y(y_1) \)).

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:
- \( t.Y = y, t_1.Y_1 = t'_1.Y_1 = y_1, t_1.Y = t'_1.Y = y' \);
- \( t_1.Y_1 = t'_1.Y^+_1, t.Y_1 = t'_1.Y^+_1 = t'_1.Y^+_m \);
- for every \( A \) not in \( Y^+_1 \), \( t_1.A \neq t'_1.A, t_1.A \neq t.A \) and \( t'_1.A \neq t.A \).

Notice that, since \( Y^+_1 \neq U \) and \( Y^+_m \neq U \), the tuples \( t, t_1 \), and \( t'_1 \) are distinct. On the other hand, using Proposition 3.1, \( \Delta' \) satisfies \( \text{FD} \) because we have \( \text{match}(t_1, t'_1) = (\text{match}(t_1, t'_1))^+ = Y^+_1 \), \( \text{match}(t_1, t'_1) = (\text{match}(t_1, t_1))^+ = Y^+_m \) and \( \text{match}(t, t'_1) = (\text{match}(t, t'_1))^+ = Y^+_1 \).

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, \( \Delta' \in \text{inst}_{\text{FD}}(\Delta, q_1) \).

Regarding the supports of \( q \) and \( q_1 \), since \( Y_m \subseteq Y, t_1.Y \) and \( t'_1.Y \) are different than \( y \), and thus, \( \text{sup}_{\Delta'}(q) = 1 \). On the other hand, as \( Y_1 \to X_1 \notin \text{FD}^+ \), we have \( X^+_1 \notin Y^+_1 \), and so \( t_1.X_1 \neq t'_1.X_1 \). Thus, \( \text{sup}_{\Delta'}(q_1) \geq 2 \), showing that \( \text{sup}_{\Delta'}(q_1) > \text{sup}_{\Delta'}(q) \), which completes the proof.

Theorem 4.2 below is a consequence of Proposition 4.3, Lemma A.1 and Lemma A.2.

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

**Proof:**
The if part of the proof is a consequence of Proposition 4.3 and Proposition 4.4. Indeed, for every table \( \Delta' \) in \( \text{inst}_{\text{FD}}(\Delta, q, q_1) \), by Proposition 4.4, \( q \leq_{\Delta'} q_1 \) implies that \( q \leq_{\Delta'} q_1 \) and so, by Proposition 4.3, \( \text{sup}_{\Delta'}(q_1) \leq \text{sup}_{\Delta'}(q) \).

Let us now turn to the only 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 \( q \not\leq_{\Delta} q_1 \) then there exists \( \Delta' \) in \( \text{inst}_{\text{FD}}(\Delta, q, q_1) \) such that \( \text{sup}_{\Delta'}(q_1) > \text{sup}_{\Delta'}(q) \).

By Definition 4.2, assuming that \( q \not\leq_{\Delta} q_1 \) implies that \( q_1 \in \Delta(U), Y_1 \to X_1 \notin \text{FD}^+ \) and one of the following statement holds: (i) \( X Y_1 \to X_1 \notin \text{FD}^+ \), or (ii) \( Y_1 \to Y \notin \text{FD}^+ \), or (iii) \( Y_1 \to Y \in \text{FD}^+ \) and \( y \notin \Delta_Y(y_1) \).

- If \( q \notin \Delta(U) \), then, for every \( \Delta' \) in \( \text{inst}_{\text{FD}}(\Delta, q, q_1) \), we have \( q \notin \Delta(U) \) and \( q_1 \in \Delta(U) \). Thus, \( \text{sup}_{\Delta'}(q) = 0 \) and \( \text{sup}_{\Delta'}(q_1) \geq 1 \). Hence, we have \( \text{sup}_{\Delta'}(q_1) > \text{sup}_{\Delta'}(q) \).

- If \( q \in \Delta(U) \), then \( q \) and \( q_1 \) are in \( \Delta(U) \). If (iii) holds, then Lemma A.2 shows that there exists \( \Delta' \) in \( \text{inst}_{\text{FD}}(\Delta, q, q_1) \) such that \( \text{sup}_{\Delta'}(q_1) > \text{sup}_{\Delta'}(q) \).

On the other hand, if (i) or (ii) holds, Lemma A.1 shows that there exists \( \Delta' \) in \( \text{inst}_{\text{FD}}(U) \) such that \( q \) and \( q_1 \) are in \( \Delta(U) \) for which we have \( \text{sup}_{\Delta'}(q_1) > \text{sup}_{\Delta'}(q) \). Moreover, if \( Y_1 \to Y \) is in \( \text{FD}^+ \), it can be seen from the proof of the lemma that the tuples \( t \) and \( t_1 \) in \( \Delta' \) are such that \( \Delta_Y(y_1) = \Delta_Y'(y_1) \).

Thus, \( \Delta' \) is in \( \text{inst}_{\text{FD}}(\Delta, q, q_1) \).

Consequently, in any of the three cases, there exists \( \Delta' \) in \( \text{inst}_{\text{FD}}(\Delta, q, q_1) \) such that \( \text{sup}_{\Delta'}(q_1) > \text{sup}_{\Delta'}(q) \). Therefore, the proof is complete.