Count Constraints for Inverse OLAP and Aggregate Data Exchange*

Domenico Saccà, Edoardo Serra, and Antonella Guzzo

DEIS, Università della Calabria, 87036 Rende, Italy
{sacca,eserra,guzzo}@deis.unical.it

Abstract. A classical problem in database theory is to verify whether there exists a relation (or database) instance satisfying a number of given dependency constraints but the issue of handling constraints on aggregate data has not been much investigated so far. This paper introduces a new type of data dependency, called count constraint, that requires the results of given count operations on a relation to be within a certain range. Count constraints are defined by a suitable extension of first order predicate calculus, based on set terms, and they are then used in a new decisional problem, the Inverse OLAP: given a fact table, does there exist an instance satisfying a set of given count constraints? Count constraints can also be used into a data exchange system context, where data from the source database are transferred to the target database using aggregate operations.

Keywords: Count Constraints, OLAP Analysis, Data Exchange

1 Introduction

A typical problem in relational database theory is to decide the existence of a database satisfying a given set of given integrity constraints. Classical approaches mainly focus on inclusion dependencies and functional dependencies [2–4]. This problem has recently received a renewed deal of interest within the context of data exchange [5–7], but the issue of handling constraints on aggregate data has not been much investigated so far, notwithstanding the relevance of aggregate operations in many applications.

In this paper we consider a new type of data dependencies, called *count constraint*, prescribing the results of given count operations on a relation to be within a certain ranges. Count constraints are relevant in *OLAP analysis*, which is characterized by multidimensional data cubes that enable manipulation and analysis of data stored in a source database from multiple perspectives in a fast way [8, 9]. In this paper we apply count constraints to a *fact table*, that is a relation scheme whose attributes are dimensions (i.e., properties, possibly structured at various levels of abstraction) and measures (i.e., values computed on the basis of some aggregation operations, e.g., count). A fact table is part of a *star schema* that typically includes also dimension tables describing dimension attributes.

* This paper is an extended abstract of [1].

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

To get an intuition of our approach, consider a fact table $\mathcal{R}(T, I)$ with two attributes T (the ID of a transaction) and I (the ID of an item) with domain $\mathcal{I}$, stored in a relation $\mathcal{D}_I$. Given a relation r on $\mathcal{R}$, $G = Group(r)By(T)$ divides r into a number of groups, one for each transaction ID. We want to express the following constraints: every item may occur in at most 1000 transactions, the itemset $i = \{a, b, c\}$ must occur as a transaction group at least 100 and at most 200 times, whereas every other set s of items cannot be present as a sub-group of more than 10 transactions, except for all subsets of i (including i) that have no limits. Such conditions can be formulated by the following count constraints, expressed with a logic formalism that extends the one adopted in the data exchange setting [5–7] – note that we write below i to represent the constant set term:

$$\forall \mathtt{I}\ (\ \mathcal{D}_{\mathtt{I}}(\mathtt{I})\ \rightarrow\ 0 \leq \#(\{\mathtt{T} : \mathcal{R}(\mathtt{T}, \mathtt{I})\}) \leq 1000\ ); \tag{1}$$

$$\forall \mathtt{s}\ (\ \mathtt{s} = i\ \rightarrow\ 100 \leq \#(\{\mathtt{T} : \mathtt{s} = \{\mathtt{I} :\ \mathcal{R}(\mathtt{T}, \mathtt{I})\}\}) \leq 200\ ); \tag{2}$$

$$\forall \mathtt{s}\,(\,\mathtt{s} \subseteq \{\mathtt{I} : \mathcal{D}_{\mathtt{I}}(\mathtt{I})\} \wedge \mathtt{s} \not\subseteq i \rightarrow 0 \leq \#(\{\mathtt{T} : \mathtt{s} \subseteq \{\mathtt{I} : \mathcal{R}(\mathtt{T}, \mathtt{I})\}\}) \leq 10\,). \tag{3}$$

Count constraints represents an extension of cardinality constraints that have been first introduced in the context of the entity-relationship model and have recently received a renewed interest. A declarative format to express them has been recently proposed in [10] with the aim of formulating more general characteristics in the process of data generation. Cardinality constraints have been also introduced within the formalism of logic programs by [11, 12]. Observe that, while constraint (1) can be easily expressed as a cardinality constraint, the other two cannot as they involve sets generated by complex grouping operations.

We use count constraints to define a new decisional problem, the *Inverse OLAP*: given a star schema consisting of one fact table, does there exist a relation instance satisfying a set of given count constraints? This problem extends the *Inverse Frequent itemset Mining* problem (`IFM` for short) [13–15] and has a potential high relevance in the contexts of generating synthetic data cubes having the same characteristics of real-world ones in terms of aggregation patterns and of privacy-preserving aggregate data exchange.

In[16, 1] we proved that the new problem is NEXP-complete under various conditions: data complexity (i.e, the number of attributes and the size of constraints are constant), program complexity (i.e, the domains are constant) and combined complexity. It is interesting to point out that the NEXP-complete results on the complexity of cardinality constraints have been detected both in [10] and in [12].

In the paper, we illustrate how count constraints can be also used in the context of data exchange. Traditionally the mapping of the data from the source to the target schema is defined by source- to-target TGDs (Tuple Generating Dependencies) and additional constraints on the target schema are specified in form of EGDs (Equality Generating Dependencies) and TGDs. We show that count constraints represent a powerful extension of EGDs.

The paper is organized as follows. In Section 2 we present the logic language for describing count constraints, introduce the inverse OLAP problem and discuss its complexity. We illustrate the usage of count constraints and the relevance of inverse OLAP in a motivating example in Section 3. In Section 4 we show how count constraints can be used into a data exchange system context. Finally in Section 5 we draw the conclusion and discuss further work.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

2 Count Constraints

Let $U = (A_1, \ldots, A_n)$ be a list of n distinct attributes on the domains $D_1, \ldots, D_n$ with given cardinalities $d_1, \ldots, d_n$. A *relation scheme* is a pair consisting of a relation name and a list of attributes. In this paper we shall only deal with exactly one relation scheme containing all attributes in U, say $\mathcal{R}(A_1, \ldots, A_n)$. We also assume that the domains of the attributes $D_1, \ldots, D_n$ are stored in suitable tables of mono-attribute relation schemes — we denote their relation schemes by $\mathcal{D}_1(A_1), \ldots, \mathcal{D}_n(A_n)$. A *relation* on $\mathcal{R}$ (also called an *instance* of $\mathcal{R}$) is any table on U.

$\mathcal{R}$ represents a *star schema* consisting of a unique *fact table* whose attributes represent dimensions - as we are only interested in count aggregation, we omit to include measures. Some of the dimensions could be organized in layers defined by Functional Dependencies (FDs) — for instance the FDs $A \rightarrow B$ and $B \rightarrow C$ state that the values of dimension A are grouped at a first level B and at a second level C. In correspondence of FDs we may have additional domain relations (usually called *dimension tables*) describing hierarchies among two dimensions, e.g., $\mathcal{D}_{A,B}$ and $D_{B,C}$.

We next introduce an extension of first order predicate calculus to define count constraints on the instances of $\mathcal{R}$. The predicate symbols are: $\mathcal{R}$, the domain relation schemes $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and possible dimension hierarchy domains. The *constants* of the language are the domain values (*domain constants*) and all (non-negative) integers.

Besides to domain constants, the Herbrand universe includes constant set terms – a set term represents a set of tuples having arity bound by some constant k. For instance, given the attributes A and B with domains $\{a_1, a_2, a_3\}$ and $\{b_1, b_2\}$ respectively, examples of constant set terms on $\{A, B\}$ are $\{[a_1, b_1], [a_2, b_1], [a_3, b_2]\}$ and $\{[a_2, b_1]\}$, while $\{[a_1], [a_3]\}$ and $\{[a_2]\}$ are two constant set terms on $\{A\}$.

A non-constant set term is defined as $\{x_1, \ldots, x_s : \alpha\}$, where $x_1, \ldots, x_s$ are variables and α is a *count formula* (defined next), in which $x_1, \ldots, x_s$ occur as free variables (similar notation for set terms and aggregate predicates has been used in the dlv system [11]). There is an interpreted function symbol $count$ (denoted by $\#$) that can be applied to a set term T to return the number of tuples in T (i.e., the cardinality of the table represented by T).

Our language is equipped with a countable number of variables and makes use of the following types of terms: (i) *simple term* (either a domain constant or a variable), (ii) *set term* (either a constant set term or a formula term) and (iii) an *integer term* (either an integer or a $count$ function term).

An *atom* can be: (i) a *relation predicate*, (ii) a *domain predicate* or (iii) a *comparison predicate* (equality or disequality of two terms, comparison of two integer terms, and inclusion of two set terms).

A *count constraint* C is a formula of one of the following two types:

1. $\forall \mathbf{X}\ (\ \phi(\mathbf{X}) \ \rightarrow\ \beta_{min} \leq \#(\{\ \mathbf{Y} : \exists \mathbf{Z}\ \psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})\ \}) \leq \beta_{max}\ )$ (*tuple count constraint*), or
2. $\forall \mathbf{X}\ (\ \phi(\mathbf{X}) \ \rightarrow\ \beta_{min} \leq \#(\ \{\ \mathbf{W} : t * \{\ \mathbf{Y} : \exists \mathbf{Z}\ \psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W})\ \}\ \}) \leq \beta_{max}\ )$ (*group count constraint*),

where $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{W}$ are disjunct lists of variables ($\mathbf{X}$ and $\mathbf{Z}$ can be empty), ϕ is a (possibly empty) conjunction of domain and comparison predicates, ψ is a conjunction

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

of relation and comparison predicates, β_{min} and β_{max} are integers for which $\beta_{min} \leq \beta_{max}$, the operator $*$ can be either $=$ or one of the two subset operators, strict inclusion ($\subset$) or possibly improper inclusion ($\subseteq$), the term t is either a constant set term or a variable in $\mathbf{X}$ that is bound to a set term in ϕ. Rule (1) of the example in the Introduction is a tuple count constraint while rules (2) and (3) are group count constraints. The formal semantics of count constraints is described in [1].

A relation r satisfies a count constraint C (and we write $r \models C$) if the evaluation of C on r is equal to true. Accordingly, r satisfies a set of count constraints $\mathbf{C}$ (and we write $r \models \mathbf{C}$) if for each $C \in \mathbf{C}$, $r \models C$.

As stated in the proposition below, checking count constraint satisfaction may require exponential time. For space reasons the proof is omitted and can be found in [1] and [16].

Proposition 1. *Given a count constraint C and a relation r on R, deciding $r \models C$ is in* EXP.

Particularly interesting are count constraints for which satisfaction can be tested in polynomial time.

Inverse OLAP Problem. *Given a set of count constraints C on R, the Inverse OLAP problem consists of deciding whether there exists a relation r on $\mathcal{R}$ such that $r \models C$.*

Proposition 2. *The Inverse OLAP problem is* NEXP*-complete.*

The proof is reported in [1] and [16]. We point out that, in our general setting, we are considering the so-called "combined complexity" [17]: both the number of attributes, the size of constraints and the domain sizes are not fixed and are part of the input. The "program complexity" version of the problem consists of considering domain sizes as constants. On the other hand, the "data complexity" version of the inverse OLAP problem fixes the number of attributes and the size of constraints and considers domain sizes as the only problem input. The NEXP-completeness of Inverse OLAP under combined complexity is not surprising: in fact, even the simple evaluation of single clause DATALOG programs is known to be EXP-complete [18]. What is really surprising is that inverse OLAP is NEXP-complete also under data complexity. This result depends mainly on group count constraints, that introduce the typical high complexity of data mining problems.

3 A Motivating Example

We refer to a classical example of point-of-sales transaction star schema. The attributes of U are: T (Transaction), I (Item), B (Brand), S (Store), A (Area) — their (finite) domains can be suitably defined. We are also given the following functional dependencies (FDs): $T \rightarrow S$, $S \rightarrow A$. It turns out that $\{T, I, B\}$ is the relation key. The domains of the attributes are denoted by $\mathcal{D}_T$, $\mathcal{D}_I$ and so on. We next present a number of meaningful count constraints that clarify their usage. To simplify the notation, all low-case letter variables are intended to be universally quantified.

(i): Enforcing FDs and relation key

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

For instance the FD $T \rightarrow S$ can be expressed as follows:

$$\mathcal{D}_{\mathtt{T}}(\mathtt{t}) \ \rightarrow \ 0 \leq \#(\{\mathtt{S} : \exists \mathtt{I}, \mathtt{B}, \mathtt{A}\, \mathcal{R}(\mathtt{t}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{A})\}) \leq 1$$

The relation key $\{T, I, B\}$ can be enforced as:

$$\mathcal{D}_{\mathtt{T}}(\mathtt{t}) \wedge \mathcal{D}_{\mathtt{I}}(\mathtt{i}) \wedge \mathcal{D}_{\mathtt{B}}(\mathtt{b}) \rightarrow 0 \leq \#(\{\mathtt{S}, \mathtt{A} : \mathcal{R}(\mathtt{t}, \mathtt{i}, \mathtt{b}, \mathtt{S}, \mathtt{A}))\}) \leq 1$$

(ii): Enforcing the overall number of tuples

There must be between 50000 and 100000 tuples in any instance of R:

$$true \rightarrow \ 50000 \leq \#(\{\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{A} : \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{A})\}) \leq 100000$$

(iii): Enforcing the total number of transactions in an area

There must be between 1000 and 2000 transactions in every region, except in "Cal" for which the upper bound is increased to 9000:

$$true \ \rightarrow 1000 \leq \#(\{\mathtt{T} : \exists \mathtt{I}, \mathtt{B}, \mathtt{S}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \text{"}\mathtt{Cal}\text{"})\}) \leq 9000;$$

$$\mathcal{D}_{\mathtt{A}}(\mathtt{a}) \wedge \mathtt{a} \neq \text{"}\mathtt{Cal}\text{"} \ \rightarrow 1000 \leq \#(\{\mathtt{T} : \exists \mathtt{I}, \mathtt{B}, \mathtt{S}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{a})\}) \leq 2000.$$

If we wish to enforce the above transaction constraint in every store of an area, we can use the dimension hierarchy domain $\mathcal{D}_{S,A}$:

$$\mathcal{D}_{\mathtt{S},\mathtt{A}}(\mathtt{s}, \text{"}\mathtt{Cal}\text{"}) \rightarrow 1000 \leq \#(\{\mathtt{T} : \exists \mathtt{I}, \mathtt{B}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{s}, \text{"}\mathtt{Cal}\text{"})\}) \leq 9000;$$

$$\mathcal{D}_{\mathtt{S},\mathtt{A}}(\mathtt{s}, \mathtt{a}) \wedge \mathtt{a} \neq \text{"}\mathtt{Cal}\text{"} \ \rightarrow 1000 \leq \#(\{\mathtt{T} : \exists \mathtt{I}, \mathtt{B}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{s}, \mathtt{a})\}) \leq 2000\ .$$

(iv): 1-arity group count constraints

Both the set of items $i = \{[a], [b], [c]\}$ and $j = \{[b], [c], [d]\}$) must be present in at least 100 and in at most 200 transactions, whereas every other set s of items cannot appear in more than 15 transactions if s contains more than 10 elements or 20 otherwise, except for all subsets of i and j that have no limits (for space reasons we write below i and j to represent the two constant set terms):

$$\mathtt{x} = i \vee \mathtt{x} = j \rightarrow 100 \leq \#(\{\mathtt{T} : \mathtt{x} \subseteq \{\mathtt{I} : \exists \mathtt{B}, \mathtt{S}, \mathtt{A}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{A})\}\}) \leq 200;$$

$$\mathtt{x} \subseteq \{\mathtt{I} : \mathcal{D}_{\mathtt{I}}(\mathtt{I})\} \wedge \mathtt{x} \not\subset i \wedge \mathtt{x} \not\subset j \wedge \#(\mathtt{x}) \leq 10 \ \rightarrow$$
$$0 \leq \#(\{\mathtt{T} : \mathtt{x} \subseteq \{\mathtt{I} : \exists \mathtt{B}, \mathtt{S}, \mathtt{A}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{A})\}\}) \leq 20;$$

$$\mathtt{x} \subseteq \{\mathtt{I} : \mathcal{D}_{\mathtt{I}}(\mathtt{I})\} \wedge \mathtt{x} \not\subset i \wedge \mathtt{x} \not\subset j \wedge \#(\mathtt{x}) > 10 \ \rightarrow$$
$$0 \leq \#(\{\mathtt{T} : \mathtt{x} \subseteq \{\mathtt{I} : \exists \mathtt{B}, \mathtt{S}, \mathtt{A}\, \mathcal{R}(\mathtt{T}, \mathtt{I}, \mathtt{B}, \mathtt{S}, \mathtt{A})\}\}) \leq 15.$$

Note that the first of the above constraints has a disjunction in the left hand side: it is only a shorthand to represent two constraints having the same right hand side.

The above constraints define an instance of an IFM problem [13–15] for which, in addition to fixing support constraints for a number of pre-defined itemsets (typically the frequent ones, in this case i and j), there are generic support constraints for all other itemsets (the unfrequent ones). The example confirms that Inverse OLAP is a powerful extension of IFM.

(v): 2-arity group count constraints

There must be at least 100 and at most 200 transactions containing an item "sm" (smartphone) of the brand "nd" (ndrangtung), whereas the same set of pairs of item and brand are sold together in at most 10 transactions, except for the ones containing the

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

pair ("sm","nd") for which the limit is 50 (for space reasons we write t to represent the singleton constant set term $\{$["sm","nd"]$\}$):

$$true \rightarrow 100 \leq \#(\{\mathtt{T} : t \subseteq \{\mathtt{I},\mathtt{B} : \exists \mathtt{S},\mathtt{A}\, \mathcal{R}(\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{S},\mathtt{A}))\}\}) \leq 200;$$

$$\mathtt{x} \subseteq \{\mathtt{I},\mathtt{B} : \mathcal{D}_{\mathtt{I}}(\mathtt{I}) \wedge \mathcal{D}_{\mathtt{B}}(\mathtt{B})\} \wedge t \subset \mathtt{x} \rightarrow$$
$$0 \leq \#(\{\mathtt{T} : \mathtt{x} \subseteq \{\mathtt{I},\mathtt{B} : \exists \mathtt{S},\mathtt{A}\, \mathcal{R}(\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{S},\mathtt{A}))\}\}) \leq 50;$$

$$\mathtt{x} \subseteq \{\mathtt{I},\mathtt{B} : \mathcal{D}_{\mathtt{I}}(\mathtt{I}) \wedge \mathcal{D}_{\mathtt{B}}(\mathtt{B})\} \wedge t \not\subset \mathtt{x} \rightarrow$$
$$0 \leq \#(\{\mathtt{T} : \mathtt{x} \subseteq \{\mathtt{I},\mathtt{B} : \exists \mathtt{S},\mathtt{A}\, \mathcal{R}(\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{S},\mathtt{A}))\}\}) \leq 10.$$

(Recall that $\subset$ denotes strict subset relationship.) The above constraints define an instance of an Inverse Frequent Itemset problem in which classical itemsets are replaced by sets of object pairs.

4 A Step towards Aggregate Data Exchange

Data exchange [5–7] is the problem of migrating a data instance from a source schema to a target schema such that the materialized data on the target schema satisfies the integrity constraints specified by it. The classical data exchange setting is: $(S, T, \Sigma_{st}, \Sigma_t)$, where S is the source relational database schema, T is the target schema, Σ_t are dependencies on the target scheme T and Σ_{st} are source-to-target dependencies.

The dependencies in Σ_{st} mapping data from the source to the target schema are TGDs (Tuple Generating Dependencies) and have the following format: $\forall \mathbf{X}(\, \phi_S(\mathbf{X}) \rightarrow \exists \mathbf{Y}\ \ \psi_T(\mathbf{X}, \mathbf{Y})\,)$, where $\phi_S(\mathbf{X})$ and $\psi_T(\mathbf{X}, \mathbf{Y})$ are formula on S and T, respectively, and $\mathbf{X}$, $\mathbf{Y}$ are lists of variables.

Dependencies in Σ_t specify constraints on the target schema, which the imported data must satisfy, and can be either TGDs or EGDs (Equality Generating Dependencies) having the form $\forall \mathbf{X}(\, \psi_T(\mathbf{X}) \rightarrow x_1 = x_2\,)$, where x_1 and x_2 are variables in $\mathbf{X}$.

It is easy to see that a generic EGD can be formulated by the following count constraint: $\forall \mathbf{X}'\ (\, true \rightarrow 0 \leq \#(\{y : \psi(y, \mathbf{X}')\}\,) \leq 1)$, where $\mathbf{X}'$ contains all variables in $\mathbf{X}$ except x_1 and x_2; moreover, y replaces both x_1 and x_2 in ψ.

Aggregate data exchange for preserving privacy

The target relational database scheme consists of a unique relation scheme, that is the one used in Section 3: $\mathcal{R}(T, I, B, S, A)$. Recall that the meaning of the attributes is: T (Transaction), I (Item), B (Brand), S (Store), A (Area), and that the following FDs hold: $T \rightarrow S$, $S \rightarrow A$.

The source relational database scheme consists of three relation schemes: $\mathcal{TR}(T, I, B)$, $\mathcal{ST}(T, S)$ and $\mathcal{AR}(S, A)$. Observe that this scheme is the normalized version of the target scheme.

We want the target relation to be the natural join of the source relation but, for privacy reasons, the associations between transactions and pairs of item and brand must be perturbed: the transactions IDs of the same store are permuted. For instance, if the store s has n transactions $t_1, \ldots, t_n$, the block of item-brand pairs of a transaction t_i are moved to a transaction t_j, then the block of t_j is moved to another transaction and so on.

Let us first use the classical setting to implement two natural joins of the source relations instead of only one so that we can later perform a permutation of transactions inside every store:

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

$$\mathcal{TR}(\mathtt{t},\mathtt{i},\mathtt{b}) \wedge \mathcal{ST}(\mathtt{t},\mathtt{s}) \wedge \mathcal{AR}(\mathtt{s},\mathtt{a}) \rightarrow \exists \mathtt{T}\, \mathcal{R}(\mathtt{T},\mathtt{i},\mathtt{b},\mathtt{s},\mathtt{a});$$
$$\mathcal{ST}(\mathtt{t},\mathtt{s}) \rightarrow \exists \mathtt{I},\mathtt{B},\mathtt{A}\, \mathcal{R}(\mathtt{t},\mathtt{I},\mathtt{B},\mathtt{s},\mathtt{A}).$$

We now use a count constraint to enforce that the total number of tuples in $\mathcal{TR}$ is equal to the total number of tuples in $\mathcal{R}$ so that the target relation cannot store additional tuples:

$$\mathtt{x} = \#(\{\mathtt{T},\mathtt{I},\mathtt{B} : \mathcal{TR}(\mathtt{T},\mathtt{I},\mathtt{B})\}) \rightarrow \mathtt{x} = \#(\{\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{S},\mathtt{A} : \mathcal{R}(\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{S},\mathtt{A})\}).$$

So we have lost the correspondence between transactions in the source and in the target scheme; but the following constraint imposes that the original structure of transactions is preserved modulo permutation of transactions IDs:

$$\mathcal{ST}(\mathtt{t},\mathtt{s}) \wedge \mathtt{x} = \{\mathtt{I},\mathtt{B} : \mathcal{TR}(\mathtt{t},\mathtt{I},\mathtt{B})\}) \rightarrow \exists \mathtt{T}\, (\, \mathtt{x} = \{\mathtt{I},\mathtt{B} : \exists \mathtt{A}\, \mathcal{R}(\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{s},\mathtt{A})\}\,).$$

Data exchange to an OLAP scheme

We now assume that the relation $\mathcal{R}(T, I, B, S, A)$ represents the source scheme. The target scheme is an OLAP scheme $\mathcal{SN}(S, I, B, N)$ that, for every store, represents in N the total number of item-brand pairs that are in all transactions of that store.

We aggregate data in the target relation using a count predicate in the following constraint:

$$\mathtt{n} = \#(\{\mathtt{T} : \mathcal{R}(\mathtt{T},\mathtt{i},\mathtt{b},\mathtt{s},\mathtt{a})\}) \rightarrow \mathcal{SN}(\mathtt{s},\mathtt{i},\mathtt{b},\mathtt{n}).$$

A final count constraint imposes that the target relation cannot store additional tuples:

$$\mathtt{x} = \#(\{\mathtt{S},\mathtt{I},\mathtt{B} : \exists \mathtt{T},\mathtt{A}\, \mathcal{R}(\mathtt{T},\mathtt{I},\mathtt{B},\mathtt{S},\mathtt{A})\}) \rightarrow \mathtt{x} = \#(\{\mathtt{S},\mathtt{I},\mathtt{B} : \exists \mathtt{N}\, \mathcal{SN}(\mathtt{S},\mathtt{I},\mathtt{B},\mathtt{N})\}\,).$$

5 Conclusion

We have introduced a new type of constraints, called count constraints, and a new inverse mining problem, called Inverse OLAP, that is a powerful extension of Inverse Frequent itemsets Mining: given a star schema consisting of a unique fact table and a number of count constraints, does there exist a satisfying relation? The new problem turns out to be NEXP complete under various conditions: combined complexity, program complexity and data complexity. We have also shown that count constraints can be used for performing aggregate data exchange.

We conclude by mentioning that, despite the high complexity of the Inverse OLAP problem, an approximate solution can be found in a limited amount of time in some practical situations even for large instances, by adopting and extending classical techniques used for solving large-scale linear programming. In [19] one of such techniques, called column-generation linear programming, is applied to solve an IFM problem (indeed non just the decision problem but the actual construction of a satisfying transaction database), that can be though of as a special case of inverse OLAP. Such techniques are capable of handling instances with several hundreds of items. In addition our current research is devoted to single out cases for which complexity of Inverse OLAP becomes polynomial under the data complexity by considering particular cases of count constraints.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

References

1. Saccà, D., Serra, E., Guzzo, A.: Count constraints and the inverse olap problem: Definition, complexity and a step toward aggregate data exchange. In: FoIKS. (2012) 352–369
2. Zhang, X., Ozsoyoglu, Z.M.: Implication and referential constraints: A new formal reasoning. IEEE Trans. on Knowledge and Data Engineering **9** (1997) 894–910
3. Rosati, R.: On the decidability and finite controllability of query processing in databases with incomplete information. In: PODS. (2006) 356–365
4. Cosmadakis, S.S., Kanellakis, P.C., Vardi, M.Y.: Polynomial-time implication problems for unary inclusion dependencies. J. of the ACM **37** (1990) 15–46
5. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. In: In ICDT. (2003) 207–224
6. Arenas, M., Barcel, P., Fagin, R., Libkin, L.: Locally consistent transformations and query answering in data exchange. In: PODS'04. (2004) 229–240
7. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core. ACM Trans. Database Syst. **30**(1) (2005) 174–210
8. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Record **26**(1) (1997) 65–74
9. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies. Mac Graw Hill (2009)
10. Arasu, A., Kaushik, R., Li, J.: Data generation using declarative constraints. In: Proceedings of the 2011 international conference on Management of data. SIGMOD '11, New York, NY, USA, ACM (2011) 685–696
11. Faber, W., Pfeifer, G., Leone, N., Dell'Armi, T., Ielpa, G.: Design and implementation of aggregate functions in the dlv system. TPLP **8**(5-6) (2008) 545–580
12. Syrjänen, T.: Logic Programs and Cardinality Constraints: Theory and Practice. Doctoral dissertation, TKK Dissertations in Information and Computer Science TKK-ICS-D12, Helsinki University of Technology, Department of Information and Computer Science (2009)
13. Mielikainen, T.: On inverse frequent set mining. In Society, I.C., ed.: Proc. of 2nd Workshop on Privacy Preserving Data Mining (PPDM). (2003) 18–23
14. Calders, T.: Computational complexity of itemset frequency satisfiability. In: PODS. (2004) 143–154
15. Calders, T.: The complexity of satisfying constraints on databases of transactions. Acta Inf. **44**(7-8) (2007) 591–624
16. Saccà, D., Serra, E., Guzzo, A.: Appendix to [1]. In: http://sacca.deis.unical.it/#view=object&format=object&id=960/gid=160. (2012)
17. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: STOC. (1982) 137–146
18. Gottlob, G., Papadimitriou, C.H.: On the complexity of single-rule datalog queries. In: LPAR. (1999) 201–222
19. Guzzo, A., Moccia, L., Saccà, D., Serra, E.: Solving inverse frequent itemset mining with infrequency constraint via large-scale linear programs. In: http://sacca.deis.unical.it/#view=object&format=object&id=981/gid=160. (2011)

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,