Consistent Queries Over Databases with Integrity Constraints

Sergio Greco
DEIS Università della Calabria, Italy

Ester Zumpano
DEIS Università della Calabria, Italy

INTRODUCTION

The aim of data integration is to provide a uniform integrated access to multiple heterogeneous information sources, which were designed independently for autonomous applications and whose contents are strictly related.

Integrating data from different sources consists of two main steps: the first in which the various relations are merged together and the second in which some tuples are *removed* (or *inserted*) from the resulting database in order to satisfy integrity constraints.

There are several ways to integrate databases or possibly distributed information sources, but whatever, integration architecture we choose, the heterogeneity of the sources to be integrated, causes subtle problems. In particular, the database obtained from the integration process may be inconsistent with respect to integrity constraints, that is, one or more integrity constraints are not satisfied. Integrity constraints represent an important source of information about the real world. They are usually used to define constraints on data (functional dependencies, inclusion dependencies, etc.) and have, nowadays, a wide applicability in several contexts such as semantic query optimization, cooperative query answering, database integration and view update.

Since, the satisfaction of integrity constraints cannot generally be guaranteed, if the database is obtained from the integration of different information sources, in the evaluation of queries, we must compute answers which are consistent with the integrity constraints. The following example shows a case of inconsistency.

Example 1. Consider the following database schema consisting of the single binary relation *Teaches (Course, Professor)* where the attribute *Course* is a key for the relation. Assume there are two different instances for the relations *Teaches*, D1={(c1,p1),(c2,p2)} and D2={(c1,p1),(c2,p3)}.

The two instances satisfy the constraint that *Course* is a key but, from their union we derive a relation which does not satisfy the constraint since there are two distinct tuples with the same value for the attribute *Course*.

In the integration of two conflicting databases simple solutions could be based on the definition of preference criteria such as a partial order on the source information or a majority criteria (Lin and Mendelzon, 1996). However, these solutions are not generally satisfactory and more useful solutions are those based on 1) the computation of 'repairs' for the database, 2) the computation of consistent answers (Arenas et al., 1999).

The computation of repairs is based on the definition of minimal sets of insertion and deletion operations so that the resulting database satisfies all constraints. The computation of consistent answers is based on the identification of tuples satisfying integrity constraints and on the selection of tuples matching the goal.For instance, for the integrated database of *Example 1*, we have two alternative repairs consisting in the deletion of one of the tuples *(c2,p2)* and *(c2,p3)*. The consistent answer to a query over the relation *Teaches* contains the unique tuple *(c1,p1)* so that we don't know which professor teaches course *c2*.

Therefore, it is very important, in the presence of inconsistent data, to compute the set of consistent answers, but also to know which facts are unknown and if there are possible repairs for the database.

TECHNIQUES FOR QUERYING AND REPAIRING DATABASES

Recently, there have been several proposals considering the integration of databases as well as the computation of queries over inconsistent databases. Most of the techniques work for restricted from of constraints and only recently have there been proposals to consider more general constraints. In this the following we give an informal description of the main techniques proposed in the literature.

- In Agarwal et al. (1995) it is proposed an extension of relational algebra, called *flexible algebra*, to deal with data having tuples with the same value for the key attributes and conflicting values for the other attributes. The technique only considers constraints defining functional dependencies and it is sound only for the class of databases having dependencies determined by a primary key consisting of a single attribute.
- In Dung (1996) it is proposed the Integrated Relational Calculus, an extension of flexible algebra for other key functional dependencies based on the definition of *maximal consistent subsets* for a possibly inconsistent database. Dung proposed extending relations by also considering null values denoting the absence of information with the restriction that tuples cannot have null values for the key attributes. The Integrated Relational Calculus overcomes some drawbacks of the flexible relational algebra. Anyhow as both techniques consider restricted cases the computation of answers can be done efficiently.
- In Lin and Mendelzon (1996), it is proposed an approach taking into account the majority view of the knowledge bases in order to obtain a new relation which is consistent with the integrity constraints. The technique proposes a formal semantics to merge first-order theories under a set of constraints.

Example 2. Consider the following three relation instances which collect information regarding author, title and year of publication of papers:

- Bib1={(John,T1,1980),(Mary,T2,1990)},
- Bib2={(John,T1,1981),(Mary,T2,1990)},
- Bib3={(John,T1,1981), (Frank,T3,1990)}

From the integration of the three databases Bib1, Bib2 and Bib3 we obtain the database Bib={(John,T1,1980), (Mary,T2,1990), (Frank,T3,1990)}.

Thus, the technique, proposed by Lin and Mendelson, removes the conflict about the year of publication of the paper T1 written by the author John observing that two of the three source databases, that have to be integrated, store the value 1980; thus the information that is maintained is the one which is present in the majority of the knowledge bases.

However, the 'merging by majority' technique does not resolve conflicts in all cases since information is not always present in the majority of the databases and, therefore, it is not always possible to choose between alternative values. Thus, generally, the technique stores disjunctive information and this makes the computation of answers more complex (although the computation becomes efficient if the 'merging by majority' technique can be applied); moreover, the use of the majority criteria involves discarding inconsistent data, and hence the loss of potentially useful information.

- In Arenas et al. (1999) it is introduces a logical characterisation of the notion of consistent answer in a possibly inconsistent database. The technique is based on the computation of an equivalent query $T_\omega(Q)$ derived from the source query Q. The definition of $T_\omega(Q)$ is based on the notion of residue developed in the context of semantic query optimization.

More specifically, for each literal B, appearing in some integrity constraint, a residue $Res(B)$ is computed. Intuitively, $Res(B)$ is a universal quantified first order formula which must be true, because of the constraints, if B is true. Universal constraints can be rewritten as denials, i.e., logic rules with empty heads of the form $\leftarrow B_1 \wedge \ldots \wedge B_n$.

Let A be a literal, r a denial of the form $\leftarrow B_1 \wedge \ldots \wedge B_n$, B_i (for some $1 \le i \le n$) a literal unifying with A and θ the most general unifier for A and B_i such that variables in A are used to substitute variables in B_i but they are not substituted by other variables. Then, the residue of A with respect to r and B_i is:

$$Res(A,r,Bi) = not((B_1 \wedge \ldots \wedge B_{i-1} \wedge B_{i+1} \wedge \ldots \wedge B_n)\ \theta\ ) = not\ B_1\theta \vee \ldots \vee not\ B_{i-1}\theta \vee not\ B_{i+1}\theta \vee \ldots \vee not\ B_n\theta.$$

The residue of A with respect to r is $Res(A,r) = \wedge_{Bi\,|A=Bi\theta}\ Res(A,r,B_i)$ consisting of the conjunction of all the possible residues of A in r whereas the residue of A with respect to a set of integrity constraints IC is $Res(A) = \wedge_{r \in IC}\ Res(A,r)$.

Thus, the residue of a literal A is a first order formula which must be true if A is true. The operator $T_w(Q)$ is defined as follows:

$T_0(Q) = Q$;

$T_i(Q) = T_{i-1}(Q) \wedge R$ where R is a residue of some literal in $Ti\text{-}1$.

The operator T_w represents the fixpoint of T.

Example 3. Consider a database D consisting of the following two relations: