Repairing and Querying Databases with Integrity Constraints

Sergio Greco
DEIS Università della Calabria, Italy

Ester Zumpano
DEIS Università della Calabria, Italy

INTRODUCTION

Data integration aims at providing a uniform integrated access to multiple heterogeneous information sources, which were designed independently for autonomous applications and whose contents are strictly related.

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. 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, \( D_1 = \{(c_1, p_1), (c_2, p_2)\} \) and \( D_2 = \{(c_1, p_1), (c_2, p_3)\} \).

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.

Obtaining consistent information from inconsistent databases is a primary issue in database management systems. 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 & 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 and (2) the computation of consistent answers (Arenas, Bertossi & Chomicki, 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 \((c_2, p_2)\) and \((c_2, p_3)\). The consistent answer to a query over the relation Teaches contains the unique tuple \((c_1, p_1)\) so that we do not know which professor teaches course \(c_2\).

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.

BACKGROUND

A (disjunctive Datalog) rule \( r \) is a clause of the form:

\[
A_1 \vee \ldots \vee A_k \leftarrow B_1, \ldots, B_m, \text{not } B_{m+1}, \ldots, \text{not } B_n, \quad k+m+n>0
\]

where \( A_1, \ldots, A_k, B_1, \ldots, B_n \) are atoms of the form \( p(t_1, \ldots, t_h) \), \( p \) is a predicate symbol of arity \( h \), and the terms \( t_1, \ldots, t_h \) are constants or variables. The disjunction \( A_1 \vee \ldots \vee A_k \) is the head of \( r \), while the conjunction \( B_1, \ldots, B_m, \text{not } B_{m+1}, \ldots, \text{not } B_n \) is the body of \( r \). We also assume the existence of the binary built-in predicate symbols (comparision operators) which can only be used in the body of rules.

An extended Datalog program extends standard Datalog programs with a different form of negation, known as classical or strong negation, which can also appear in the head of rules. An extended atom is either an atom, say \( A \) or its negation \( \neg A \). An extended Datalog program is a set of rules of the form:

\[
A_1 \vee \ldots \vee A_k \leftarrow B_1, \ldots, B_m, \text{not } B_{m+1}, \ldots, \text{not } B_n, \quad k+m+n>0
\]

where \( A_1, \ldots, A_k, B_1, \ldots, B_n \) are extended atoms.

The semantics of an extended program \( P \) is defined by considering each negated predicate symbol, say \( \neg p \),
as a new symbol syntactically different from \( p \) and by adding to the program, for each predicate symbol \( p \) with arity \( n \) the constraint \( \leftarrow p(X_1, \ldots, X_n), \neg p(X_1, \ldots, X_n) \).

### INTEGRITY CONSTRAINTS

Integrity constraints express semantic information over data, that is, relationships that must hold among data in the theory. Generally, integrity constraints, denoted as IC, represent the interaction among data and define properties which are supposed to be explicitly satisfied by all instances over a given database schema. Therefore, they are mainly used to validate database transactions.

**Definition 1.** A full (or universal) integrity constraint is a formula of the first order predicate calculus of the form:

\[
(\forall X) \left[ B_1 \land \ldots \land B_n \land \varphi \supset A_1 \land \ldots \land A_m \land \psi_1 \land \ldots \land \psi_k \right]
\]

where \( A_1, \ldots, A_m, B_1, \ldots, B_n \) are base positive literals, \( \varphi, \psi_1, \ldots, \psi_k \) are built-in literals, \( X \) denotes the list of all variables appearing in \( B_1, \ldots, B_n \) and it is supposed that variables appearing in \( A_1, \ldots, A_m, \varphi, \psi_1, \ldots, \psi_k \) also appear in \( B_1, \ldots, B_n \).

In the definition above, the conjunction \( B_1 \land \ldots \land B_n \land \varphi \) is called the body and the disjunction \( A_1 \land \ldots \land A_m \land \psi_1 \land \ldots \land \psi_k \) the head of the integrity constraint. Moreover, an integrity constraint is said to be positive if no negated literals occur in it.

### 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 form of constraints and only recently have there been proposals to consider more general constraints. In 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 consist-

- In Dung (1996), it is proposed that 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), an approach is proposed 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:

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

From the integration of the three databases \( \text{Bib1}, \text{Bib2}, \text{and Bib3} \), we obtain the database

\( \text{Bib} = \{(\text{John},T1,1980), (\text{Mary},T2,1990), (\text{Frank},T3,1990)\} \).

Thus, the technique, proposed by Lin and Mendelzon, 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 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, hence, the loss of potentially useful information.
Related Content

A Deterministic Approach to XML Query Processing with Efficient Support for Pure and Negated Containments
[www.igi-global.com/chapter/deterministic-approach-xml-query-processing/8020?camid=4v1a](www.igi-global.com/chapter/deterministic-approach-xml-query-processing/8020?camid=4v1a)

Complementing Business Process Verification by Validity Analysis: A Theoretical and Empirical Evaluation
[www.igi-global.com/article/complementing-business-process-verification-validity/55131?camid=4v1a](www.igi-global.com/article/complementing-business-process-verification-validity/55131?camid=4v1a)

A Cost-Based Range Estimation for Mapping Top-k Selection Queries over Relational Databases
[www.igi-global.com/article/cost-based-range-estimation-mapping/37210?camid=4v1a](www.igi-global.com/article/cost-based-range-estimation-mapping/37210?camid=4v1a)

Semantic Integration and Knowledge Discovery for Environmental Research
[www.igi-global.com/article/semantic-integration-knowledge-discovery-environmental/3366?camid=4v1a](www.igi-global.com/article/semantic-integration-knowledge-discovery-environmental/3366?camid=4v1a)