Chapter XXXIX
Merging, Repairing, and Querying Inconsistent Databases

Luciano Caroprese
University of Calabria, Italy

Ester Zumpano
University of Calabria, Italy

INTRODUCTION

Data integration aims to provide a uniform integrated access to multiple heterogeneous information sources designed independently and having strictly related contents. However, the integrated view, constructed by integrating the information provided by the different data sources by means of a specified integration strategy could potentially contain inconsistent data; that is, it can violate some of the constraints defined on the data. In the presence of an inconsistent integrated database, in other words, a database that does not satisfy some integrity constraints, two possible solutions have been investigated in the literature (Agarwal, Keller, Wiederhold, & Saraswat, 1995; Bry, 1997; Calì, Calvanese, De Giacomo, & Lenzerini, 2002; Dung, 1996; Grant & Subrahmanian, 1995; S. Greco & Zumpano, 2000; Lin & Mendelzon, 1999): repairing the database or computing consistent answers over the inconsistent database. Intuitively, a repair of the database consists of deleting or inserting a minimal number of tuples so that the resulting database is consistent, whereas the computation of the consistent answer consists of selecting the set of certain tuples (i.e., those belonging to all repaired databases) and the set of uncertain tuples (i.e., those belonging to a proper subset of repaired databases).

Example 1. Consider the database consisting of the relation *Employee(Name, Age, Salary)* where the attribute *Name* is a key for the relation, and suppose we have the integrated database DB = {Employee(Mary, 28, 20), Employee(Mary, 31, 30), Employee(Peter, 47, 50)}. DB is inconsistent and there are two possible repaired databases each obtained by deleting one of the two tuples whose value of the attribute *Name* is Mary. The answer to the query asking for the age of Peter is constituted by the set of certain tuples {<47>}, whereas the answer to the query asking for the age of Mary produces the set of uncertain values {<28>, <31>}.

This work proposes a framework for merging, repairing, and querying inconsistent databases. To this aim the problem of the satisfaction of integrity constraints in the presence of null values is investigated and a new semantics for constraints satisfaction, inspired by the approach presented in Bravo and Bertossi (2006), is proposed. The present work focuses on the inconsistencies of a database instance with respect to particular types of integrity constraints implemented and maintained in a commercial DBMS (database management system) such as primary keys, general functional dependencies, and foreign-key constraints.

The framework for merging, repairing, and querying inconsistent databases with functional dependencies restricted to primary-key constraints and foreign-key constraints has been implemented in a system prototype, called RAINBOW, developed at the University of Calabria.

PRELIMINARIES

Before presenting the problems related to the merging, repairing, and querying of inconsistent databases, let us introduce some basic definitions and notations. For additional material see Abiteboul, Hull, and Vianu (1994) and Ullman (1989).

A relational schema DS is a pair DS = $<R_S, IC>$ where Rs is a set of relational symbols and IC is a set of integrity constraints; that is, it is an assertion that has to be satisfied by a generic database instance. Given a database schema DS = $<R_S, IC>$ and a database instance DB over R_S, we say that DB is consistent if DB |= IC, in other words, if all integrity constraints in IC are satisfied by DB; otherwise, it is inconsistent.

A relational query (or simply a query) over R_S is a function from the database to a relation. In the following we assume queries over $<R_S, IC>$ are conjunctive queries. We will denote with Dom the database domain, that is, the set of values an attribute can assume, consisting of a possibly infinite set of constants, and assume $\perp \in$ Dom, where $\perp$ denotes the null value.

DATABASE MERGING

Once the logical conflicts owing to the schema heterogeneity have been resolved, conflicts may arise during the integration process among data provided by different sources. In particular, the same real-world object may correspond to many tuples that may have the same value for the key attributes but different values for some nonkey attribute.

The database integration problem consists of the merging of *n* databases $DB_1 = \{R_{1,1}, .., R_{1,n1}\}$, …, $DB_k = \{R_{k,1}, .., R_{k,nk}\}$. In the following we assume that relations corresponding to the same concept and furnished by different sources are homogenized with respect to a common ontology so that attributes denoting the same property have the same name (Yan & Ozsu, 1999). We say that two homogenized relations R and S, associated with the same concept, are overlapping if they have the same value for the key attributes. Given a set of overlapping relations, an important feature of the integration process is related to the way conflicting tuples are combined. Before performing the database integration, the relations to be merged must be first reconciled so that they have the same schema.

Definition 1. Given a set of overlapping relations $\{S_1,\ldots, S_n\}$, a reconciled relation R is $attr(R) = \bigcup_{1}^{n} attr(S_i) \cup \{Src\}$, and contains all tuples $t \in S_i$, $1 \leq i \leq n$. All attributes belonging to attr(R) - $attr(S_i)$ are fixed to $\perp$; R[Src] = i, where i is the unique index of the source database.

Example 2. Consider the following two overlapping relations S_1 and S_2.

K	Title	Author
1	Moon	Greg
2	Money	Jones
3	Sky	Jones

S_1

K	Title	Author	Year
3	Flower	Smith	1965
4	Sea	Taylor	1971
7	Sun	Steven	1980

S_2