Chapter X

Integrity Maintenance in Extensible Databases

Ulrich Schiel
Universidade Federal da Paraiba, Brazil

INTRODUCTION

The use of databases for advanced applications is a rapidly growing and changing field, due to the continuous incorporation of new technologies and media in current systems. Whereas in the near past Database Management Systems (DBMS) mainly use to store and manage tabular data, now they need to model complex structured objects, multimedia data, semi-structured and unstructured documents. Each of these improvements has its own semantics and complexity.

In order to allow an adequate description of database applications, data models are used to describe the conceptual schema of the database. If new categories of applications need to be incorporated or created, and the data model does not fit well with these applications, the model itself must be expanded. The semantics of the new constructs must be defined and the integrity of objects in the new constructs must be guaranteed.

Since a DBMS is in general not expandable, except for future versions of the same product, there are two alternatives: (i) to move the whole application to another system that is capable to adequately process the new structures, or (ii) to develop specific routines, probably with its own storage systems, in order to incorporate the new application. Clearly both solutions are unsatisfactory.

The first solution is only applicable if there exists a DBMS that considers the new structures. Even if it exists, moving to the new environment means reimplementation of the application, and this is very traumatic and demands
a lot of time and money. The other solution, to expand existing applications by special modules is more straightforward, but creates an unbalanced heterogeneous system, combining a databases with a file system. This generates problems of integration and does not allow a unified view of the data of the application.

Despite actual existing DBMS consider many modern database concepts, such as object-orientation and triggers, there are a lot of applications needing more. Concepts of temporal databases, geographic databases, hypertext (Web-) databases are not well attended.

In this chapter we introduce an approach of defining the semantics of a complex data model by means of general (schema-) integrity constraints integrated to the system as rules. This approach allows an easy way to define the semantics of complex data models. The rules systems can, at any time, be expanded in order to incorporate concepts of new applications and can also be used to add application specific integrity constraints. Therefore, data model specific constraints and application specific constraints are treated in a unified manner.

INTegrity in databases

Integrity maintenance in a database is achieved with two kinds of integrity constraints: implicit integrity constraints and explicit integrity constraints (Elmasri & Navathe, 1999).

Implicit integrity constraints, also denoted as schema constraints, are constraints defined in the conceptual database schema using the language of a data model, including attribute types, keys, null values, relationship cardinalities, generalizations, and aggregations. If we use a complex semantic data model to describe the conceptual schema, several implicit constraints are built in the schema, which reflects the expressiveness of the underlying data model.

Schema constraint satisfaction can be achieved mainly by three distinct approaches: (1) the DBMS supports the data model completely, and therefore its semantics is embedded in the software of the DBMS; (2) with the mapping of the conceptual schema into an internal schema, supported by the DBMS, the implicit constraints are implemented in the structure of the internal schema and, for features not foreseen in the model of the internal schema, create some controlling procedures; (3) the semantics of the data model is described in form of rules which are able to guarantee full semantic integrity and may be achieved in the DBMS. Since the rules are model dependent, and not application dependent, they are mapped only once to the internal schema.