Database Integrity

Jorge H. Doorn
INTIA, Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

INTRODUCTION

One of the factors that stimulated the development of database technology was the need to guarantee the consistency of the stored data. On the other hand, the demand to process more complex data with more semantic content have also lead to a better understanding of the properties of the data and motivated the evolution of the Data Base Management Systems (DBMS) toward the inclusion of facilities to handle such properties. The handling of data properties within the context of a database engine faces two main difficulties: 1) Properties are hard to identify, especially those obvious in the actual world; 2) Data property presentation varies during the process of design of the software.

The first one falls in the field of the Requirement Engineering (Loucopoulos & Karakostas, 1995). The proper requirements elicitation of a software product is a key factor in the success of the whole software development process. However, these requirements are not easy to deal with. They have different intrinsic natures and they may appear showing different faces. In many cases, some requirements are totally or partially hidden in the information collected by the software developers (Jackson, 1995).

The second one falls in the fields of Conceptual modeling and Earlier Logical Design (Doorn & Rivero, 2002). During these activities, issues concerning data modeling and DBMS paradigms promote changes in the way they are considered. Pragmatic issues such as the degree of adhesion of a given DBMS or others related with performance may introduce additional mutations. The understanding of the evolution from actual world data properties to database world integrity restrictions is another a key factor in the success of the software artifact during its life cycle. This understanding must be accompanied with a well defined traceability mechanism, which must clearly identify the actual world data property behind every database integrity restriction (Pinheiro, 2003). Traceability is also a key factor especially during software maintenance activities.

BACKGROUND

Not every data property must be modeled, but each property must be looked at carefully to see if modeling is needed in the context of the scope and in the objective of the software artifact. A more analytical approach may order the data properties by taking into account their importance (Karlsson, 1996).

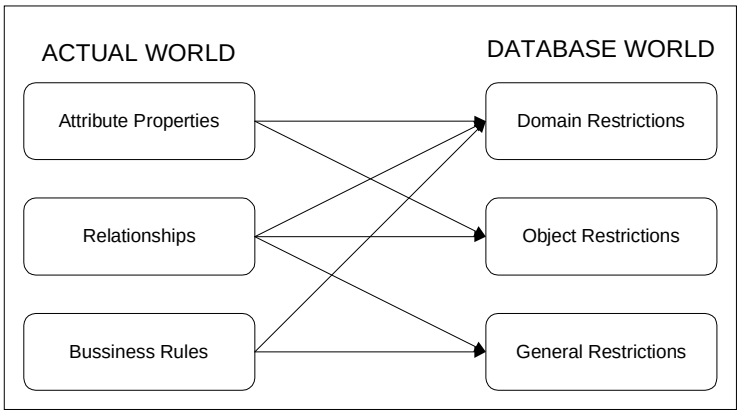
When a data property describes the allowed values for attributes, it is called Attribute Property. Another kind of data property establishes connections among different attributes, known as Relationships. When a data property carries out a semantic that is specific of the Universe of Discourse, not found in other occurrences of the same data, it is called a Business Rule (Ceri et al., 1997; Codd, 1990; Ross, 1997).

Many authors have addressed the restrictions in Databases. In Dey (1999), a profound analysis is done about ternary and higher degree relationships. The authors depict a general framework for the analysis of such constructs and provide generalized rules for their representation in the relational model. Ceri et al. (2000) analyze the order in which triggers, stored procedures, and referential integrity restrictions are executed. Regarding this matter, SQL3 (Cochrane, 1996) has defined a precise execution order and scope of its clauses. A survey of the state of the art of the semantic integrity constraints in some relational and object relational available database systems is done in Rivero et al. (2002).

EVOLUTION OF DATA PROPERTIES

Figure 1 shows all possible mappings from data properties in the actual world to database integrity restrictions. It should be noticed that there are neither arrows from Business Rules to Object Restrictions, nor from Attribute Properties to General Rules. These mappings are extremely hard to find. Also, some other mappings shown in Figure 1 occur very seldom, while others are common. Figure 2 introduces a basic taxonomy of both actual world data properties and database integrity restrictions and gives a more detailed mapping.

Figure 1. Mappings data properties into integrity restrictions



Attribute Properties. Database Engines have a set of built-in data types whose main purpose is to deal with the Attribute Properties of the data to be stored. These data types are useful and have been used for decades, helping users to take care of their data processing needs.

As with all the other Non-Functional Requirements, Attribute Properties are expressed in the Universe of Discourse in declarative ways. An Attribute Property defines the Set of Values that the attribute may have. These sets may be defined by enumeration of the members or by intention. Basically, Attribute Properties defined by enumeration map to Object Restrictions while those defined by intention may map to Domain Restrictions upon the ability of the DMBS to express the restriction (see arrows leaving Attribute Properties in figure 2). When an Attribute Property is defined by enumeration, the list of all possible values for such an attribute should be stored somewhere in the Database. It will become an inclusion dependency when such list is not key of the holding object. Otherwise it will be materialized as a referential integrity constraint.

Relationship Properties. The connection among different attributes is the source of most of the data processing richness and problems. These connections have a scope larger than Attribute Properties since they involve several attributes, at least two, usually belonging to different objects or entities.

When software artifacts are involved, the links among real world things (persons, objects, activities, etc.) are present throughout the whole process of their development. Sooner or later, the links among those things become data Relationships (see arrows leaving Relationships in figure 2). How soon this happens depends upon the software design approach. Some Relationships are binary (cardinalities may be 1:1, 1:N, N:M); however, Relationships may connect three or more real objects. When Relationships must become persistent in the data repository, an obvious issue needs to be analyzed: how are they preserved? Some very old approaches put the linked data together to express the Relationship. These approaches used to have many well-known disadvantages, especially when the Relationship cardinality was not 1:1. This redundancy was also known as the source of a new problem: consistency of data being one of the reasons that pushed towards the creation of the first database models. A Relationship whose cardinality is N:M between entities or objects introduces new problems. Firstly, there may be attributes belonging to the Relationship itself and second, more than one link is needed. The redundancy can be reduced or avoided expressing the links among data in other ways. This implies the inclusion of a special attribute in the data not found in the real world but used to represent the link. This attribute may be either a physical reference telling where the related data is stored, or a logical reference holding a key attribute that permits the related data to be found. In both cases, the redundancy problem is replaced by a referential integrity problem. To sum up, the technique used to store the data in the computer resources will create one or both of the following problems: Data Redundancy or Referential Integrity.

No matter how it is referenced, the referred data should be available every time it is needed. But since the referred data is stored and processed independently from the referring data, the link may become lost. This is called the Referential Integrity problem.

In most cases Relationships maps into Object Restrictions. However, it may happen that Relationships can be reduced to Domain Restrictions when the set of possible values is small. On the other hand, complex Relationships