Chapter I

Introduction to Database Integrity

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

Laura C. Rivero
Universidad Nacional del Centro de la Provincia de Buenos Aires and Universidad Nacional de La Plata, Argentina

Viviana E. Ferraggine
Universidad Nacional del Centro de la Provincia de Buenos Aires, Argentina

Computers are used to manage information. This use may range from collecting a small piece of data, performing a calculation and producing an output as in an embedded micro-controller application to the processing and storage of huge amounts of complex data seen in large databases. Design software to perform such information management is a difficult task. This book is oriented to discuss problems that arise in software products with significant amounts of data. Software developers have to deal with the capture and understanding of complex requirements, the design of the architecture of the software, and the development of the planned software artifacts. Along with the software product life cycle, many other activities are carried out, such as setting the software into service, training users and adapting to a changing world. The proper requirement’s elicitation of a software product is a key factor in the success of the whole process.

However, these requirements are not easy to deal with. They have different intrinsic nature 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).

This chapter appears in the book, Database Integrity: Challenges and Solutions by Jorge Doorn and Laura Rivero. Copyright © 2002, Idea Group Publishing.
More than one criterion may be used to characterize requirements. One that seems to be useful is to divide them into those properties related to what the software has to do, and those properties that the software should have. The first group is usually called Functional Requirements and the second Non Functional Requirements. A Functional Requirement could be, for example, “The system should present the current balance of the customer account in the screen” and a related Non Functional Requirement may be, “The customer balance should be ready in less than five seconds”.

In most cases, Non Functional Requirements are harder to perceive and model than Functional Requirements. Functional Requirements are usually expressed as procedures, methods or activities related to the software behavior. On the other hand, Non Functional Requirements are expressed as rules or properties that must be satisfied in a more declarative way.

Non Functional Requirements cover different areas of the desired product such as security, performance and output quality. The data to be stored and processed by the software have properties that must be ensured. Examples of data properties are found everywhere; however, they are usually disregarded. This occurs because most data properties are obvious and everybody knows about them, but also mostly everybody forgets them. For example, there is no need to say that the age of one person is always younger than the age of his or her parents. Everybody knows it, but the Database Engine where this data is stored does not (Loucopoulos & Karakostas, 1995).

The data and procedures approach to software design have to deal with the problem of the properties of the data. The object oriented approach works with objects and relationships among them. It seems that there is no data involved in this analysis. Actually, the problem from the data properties point of view is a little worse since the attributes of involved objects have properties and maybe these properties relate one object attribute with another object attribute. This is because the object orientation watches the Universe of Discourse using a model that has an extra layer between the developer and the data properties. This disadvantage does not damage the advantages of the object-oriented approach; however, the developer has to cope with it.

Not every data property must be modeled; it must be looked at carefully to see if it 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, taking into account their importance (Karlsson, 1996).

When a data property describes the allowed values for attributes, it is called Domain Property. Another kind of data properties establishes connections among different attributes; these are known as Relationships. When a data property carries out a semantic that is specific to the Universe of