Chapter XIX
Object–Relational Modeling

Jaroslav Zendulka
Brno University of Technology, Czech Republic

INTRODUCTION

Modeling techniques play an important role in the development of database applications. Well-known entity-relationship modeling and its extensions have become a widely-accepted approach for relational database conceptual design. An object-oriented approach has brought a new view of conceptual modeling. A class as a fundamental concept of the object-oriented approach encapsulates both data and behavior, whereas traditional relational databases are able to store only data. In the early 1990s, the difference between the relational and object-oriented (OO) technologies, which were, and are still used together to build complex software systems, was labeled the *object-relational impedance mismatch* (Ambler, 2003).

The object-oriented approach and the need of new application areas to store complex data have greatly influenced database technology since that time. Besides appearance of object-oriented database systems, which fully implement object-oriented paradigm in a database environment (Catell et al., 2003), traditional relational database management systems become object-relational (Stonebraker & Brown, 1999). The most recent versions of the SQL standard, SQL: 1999 (Melton & Simon (2001) and SQL: 2003 (Eisenberg et al., 2004), introduced object-relational features to the standard and leading database producers have already released packages which incorporate them.

Development of complex data intensive software systems involves a close working relationship between software and database developers. Currently, software developers deal with object-

oriented software development and use object-oriented modeling techniques to represent the main view of the application, whereas database developers model, design, build, and optimize the database. However, modeling techniques for relational databases do not support important features of object-relational databases. In addition, shared vision and clear communication of project details are necessary conditions for a software project to be successful. A common modeling language and supporting development tools can provide good conditions for it.

The Unified Modeling Language (UML) was adopted as an OMG (Object Management Group) standard for object modeling in 1997. Since that time, it has become popular and widely used. It provides several modeling techniques (diagrams) that visualize a system from different perspectives. From database design point of view, a class diagram is the most important diagram—it shows a set of structural elements and their static relationships. It can be used for conceptual modeling of persistent classes. But the UML does not contain direct support neither for capturing important features of relational databases, nor for specific features of object-relational databases that are necessary for modeling data stored in a relational database and objects stored in an object-relational database at design levels below the conceptual one. Therefore, it is necessary to add the ability to model features of this kind of databases in an effective and intelligible way. Fortunately, the UML provides an extensibility mechanism that allows doing it.

This article shows how an object-relational database schema can be modeled in UML. This technique will be referred to as object-relational modeling here.

BACKGROUND

In this section, we summarize important features of the SQL:1999 object-relational model (Melton & Simon 2001, Database Language SQL, 1999, Badia 2006) with some extensions available in SQL:2003 (Eisenberg et al., 2004, Database Language SQL, 2004). We also explain how the model is implemented by Oracle in their database servers. Then we describe extensibility mechanisms of the UML, which we exploit in the next sections.

The Object-Relational Data Model in Standard SQL and in Oracle SQL

We can say that since SQL:1999 the underlying model of the standard is object-relational and therefore the standard has also become a standard database language for object-relational databases. We only focus on the most important features of the model that represent object extensions here.

The model keeps the concept of tables but it removes the First Normal Form (1NF), which is the basic requirement of the relational model of data. The standard introduces *user-defined types* (UDTs), which mean types specified by users. There are two types of UDTs – distinct types and structured ones. The *structured types* are more interested from our point of view. A structured type can have an internal structure referred to as attributes and it can include methods that represent behavior of instances of the type. Similarly to classes, a structured type can inherit properties of another type. A structured type can represent a domain of a table column or it can specify the type of tupples in a table. There is an important difference concerning value identity between these two cases. Values in the former case do not have any identity. They are values of a composite attribute. The tupples in the latter case have identity known from the OO approach as an object identifier (OID). They represent objects with corresponding attributes and methods, and the table that stores them is said to be a *typed table*. The SQL refers the identifying values to as *REF values* and the relevant type as *REF type*. It is possible to specify the domain of