Object–Relational Modeling in the UML

Jaroslav Zendulka
Brno University of Technology, Czech Republic

INTRODUCTION

Modeling techniques play an important role in the development of database applications. One of the trends in current database management systems is that they become object-relational (Stonebraker & Brown, 1999). The most recent version of the SQL standard, SQL:1999, includes object-relational features, and a number of leading companies have already released packages that incorporate them.

Well-known modeling techniques for relational databases, such as entity-relational diagrams, do not support important features of object-relational databases. In addition, the development of a database application involves a close working relationship between the software and database developers. Software developers deal with object-oriented software development and use object-oriented modeling techniques, such as a logical class model, to represent the main view of the application, whereas database developers model, design, build, and optimize the database. The most successful projects are marked by a shared vision and clear communication of project details (IBM, 2001). A common modeling language and supporting development tools can provide good conditions for it.

The Unified Modeling Language (UML) was adopted as an Object Management Group (OMG) standard for object modeling in 1997. Since that time, it has become popular and widely used and provides several types of 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. This model can be used not only as documentation, but also for data definition language (DDL) statements generation. If we want to employ the UML as a modeling language for development of a database application where persistent data is stored in an object-relational database, it is necessary to add the ability to model features of these kinds of databases in an effective and intelligible way, and the UML provides a proper extension mechanism for it. Modeling of an object-relational database schema will be called object-relational modeling in this article.

If we accept three perspectives for drawing class diagrams (Fowler, 2003)—conceptual, specification, and implementation—the need for object-relational modeling is only for the implementation, and possibly specification, levels. A conceptual model should be developed with little or no regard to implementation and a target database environment.

BACKGROUND

The UML provides three extensibility mechanisms that make it possible to extend the language in controlled ways (Booch, Rumbaugh, & Jacobson, 1998; Object Management Group [OMG], 2003):

- stereotypes,
- tagged values, and
- constraints.

A stereotype extends a vocabulary of the UML. It allows the introduction of a new model element derived from one existing in the UML metamodel. A tagged value extends the properties of the UML’s model elements. It is a keyword-value pair element that may be attached to any kind of a model element. The keyword is called a tag. A constraint extends the semantics of a model block by means of specifying conditions and propositions that must be maintained as true, otherwise the system described by the model is invalid. There are some standard stereotypes, tagged values, and constraints predefined in the UML. One of them is a stereotype <<Table>>, which is a stereotype of the UML class element.

The main purpose of the extendability mechanisms is to tailor the UML to the specific needs of a given application domain or target environment. It makes it possible to develop a predefined set of stereotypes, tagged values, and constraints, and notation icons that collectively specialize and tailor the UML for specific domain or process. Such a set is called a profile. Several profiles has already been accepted by OMG as standard profiles, but none of them is for data or object-relational modeling.

Several works that propose extensions of the UML for data and object-relational modeling has been presented. Most of the extensions have been proposed not for SQL:1999 but for Oracle8, because this database management system (DBMS) had provided object extensions...
before SQL:1999 was published. Probably the most im-
portant proposal has been developed and implemented by
Rational Software Corporation in their Rational Rose
product, which is one of the best-known UML-oriented
modeling tools. It provides support not only for object-
oriented database modeling but also for relational data-
base modeling (Rational Software Corp., 2001) and Oracle8
object-relational modeling. The Rose Oracle8 tool permits
both forward and backward engineering of Oracle8 ob-
ject-relational schemas.

Marcos, Vela, and Cavero (2001, 2003) proposed sev-
eral stereotypes, tagged values, and constraints for the
modeling of structured types, typed tables, ARRAY type,
REF type, two types of methods in the SQL:1999, and for
modeling of similar elements in Oracle8i.

The approach presented in this article is based on
extensions for Oracle8 by Rational Software, but it can
be used for SQL:1999, too. Examples are drawn in
Rational Rose, with the Rose Oracle8 tool.

OBJECT-RELATIONAL MODEL IN
ORACLE8 AND SQL:1999

Both the SQL:1999 and the SQL dialect of Oracle8 (and
more recent releases) extend the relational model in
several important directions. Only those modeling fea-
tures that are presented in this article are summarized.
More on new features of the SQL:1999 can be found in
Melton & Simon (2001) or in the standard specification
(Database Language SQL, 1999). More information on
the Oracle object-relational model is available in Oracle
documentation (Oracle, 2003a, 2003b).

First, both Oracle8 and SQL:1999 relinquished the
basic demand on the relation in the relational model—
to be in the 1NF. The user can define user-defined data
types. In Oracle, there are two types of user-defined
data types: object types and collection types.

An object data type is an abstraction of a real-world
entity, the representation of which will be stored in a
database. An object type is a schema object with a name,
set of attributes and methods. Each attribute is of either
a built-in scalar data type or a user-defined data type.
This allows for the defining object types with a complex
data structure.

Methods of an object data type implement opera-
tions with the data type. Every object type has a system-
defined constructor method.

An object data type is a template for objects. Objects
can be instantiated by the constructor method of a given
object type and stored in object tables. An object table
can be viewed either as a single column table of row
objects or as a multicolunm table. Each row object has
assigned a unique object identifier (OID). It can be sys-
tem-generated or primary-key based. Oracle provides a
built-in data type called REF to encapsulate references to
row objects. This type can be used to implement links
between objects.

There are two collection types available: array types
and table types.

Both collection types are sets of data elements of
the same type, but there are important differences be-
tween them. An array type (called VARRAY) is an
ordered and bounded set, whereas a table type (called a
nested table) is unordered and unbounded. In addition,
a VARRAY data value is stored and retrieved as one data
unit, whereas a nested table value is stored in a storage
table with every element of the collection mapped into
a row of the storage table.

Another object extension concerns views. Just as a
relational view is a virtual table, an object view is a
virtual object table. Using object views, it is possible to
create virtual object tables with columns of both built-in
and user-defined data types mapped to columns of
relational or object base tables. Object views provide
the ability to offer specialized or restricted access to
data stored in relational and object tables. In addition,
they provide the ability to view relational data as ob-
jects. The object view definition contains information
about the object type of the view objects, the way of
constructing OID, and the mapping SELECT statement.

The Oracle8 object-relational model can be described
as a metamodel in the UML (Zendulka, 2001). Although
most of Oracle8 extensions are included in SQL:1999,
there are some differences. First, the terminology of
SQL:1999 differs from that of Oracle8 (e.g., in
SQL:1999, object data types are called structured types
and object tables/views are referred to as referenceable
tables/views). In addition, only array as a collection
type is available in SQL:1999. On the other hand, Oracle8
does not support inheritance (more recent versions do)
whereas SQL:1999 does.

EXTENSION OF THE UML

The mapping used in Rational Rose Oracle8 can be
perceived as a profile definition for object-relational
modeling with Oracle8 as a target DBMS. The profile
introduces several stereotypes, some constraints in a
form of conventions, and tagged values. The tagged
values have a form of schema-generation properties
attached to a project, class, operation, and attribute.
They contain such values as, for example, a WHERE
clause of a view definition. Stereotypes of the profile
are listed in Table 1.
Related Content

Matching Attributes across Overlapping Heterogeneous Data Sources Using Mutual Information
www.igi-global.com/chapter/matching-attributes-across-overlapping-heterogeneous/63676?camid=4v1a

Enhancing the Process of Knowledge Discovery in Geographic Databases Using Geo-Ontologies
www.igi-global.com/chapter/enhancing-process-knowledge-discovery-geographic/8044?camid=4v1a

Application of Mobile Agents in Mobile Data Access Systems: A Prototype
www.igi-global.com/article/application-mobile-agents-mobile-data/3318?camid=4v1a

Query Processing in Spatial Databases
www.igi-global.com/chapter/query-processing-spatial-databases/20711?camid=4v1a