Regular paper

A Methodological Approach for Object-Relational Database Design using UML

Esperanza Marcos, Belén Vela, José María Cavero

Kybele Research Group, Rey Juan Carlos University, 28933 Madrid, Spain; E-mail: {cuca,b.vela,j.m.cavero}@escet.urjc.es

Initial submission: 22 January 2002 / Revised submission: 10 June 2002
Published online: 7 January 2003 – © Springer-Verlag 2003

Abstract. The most common way of designing databases is by means of a conceptual model, such as E/R, without taking into account other views of the system. New object-oriented design languages, such as UML (Unified Modelling Language), allow the whole system, including the database schema, to be modelled in a uniform way. Moreover, as UML is an extendable language, it allows for any necessary introduction of new stereotypes for specific applications. Proposals exist to extend UML with stereotypes for database design but, unfortunately, they are focused on relational databases. However, new applications require complex objects to be represented in complex relationships, object-relational databases being more appropriate for these requirements. The framework of this paper is an Object-Relational Database Design Methodology, which defines new UML stereotypes for Object-Relational Database Design and proposes some guidelines to translate a UML conceptual schema into an object-relational schema. The guidelines are based on the SQL:1999 object-relational model and on Oracle8i as a product example.


1 Introduction

In spite of the impact of relational databases over the last few decades, databases of this kind have some limitations for supporting the data persistence required by present day applications. Owing to recent hardware improvements, more sophisticated applications have emerged, such as CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing), CASE (Computer-Aided Software Engineering), GIS (Geographic Information System), etc. They may be characterized as consisting of complex objects with complex relationships. Representing such objects and relationships in the relational model implies that the objects must be broken down into a large number of tuples. Thus, a considerable number of joins are necessary to retrieve an object and, when tables are too deeply nested, performance is significantly reduced [3]. A new generation of databases has appeared to solve these problems: the object-oriented database generation, which includes object-relational [24] and object databases [4]. This new technology is well suited for storing and retrieving complex data because it supports complex data types and relationships, multimedia data, inheritance, etc.

Nonetheless, good technology is not enough to support complex objects and applications. It is necessary to define methodologies that guide designers in the object database design task, in the same way as has traditionally been done with relational databases. In recent years some approaches to object-oriented database design have appeared [5,11,18,23,24]. Unfortunately, none of these proposals can be considered “the method”, either for object-relational or for object databases. On the one hand, they do not consider the latest versions of the representative standards for the two technologies: ODMG 3.0 for object databases [8] and SQL:1999 for object-relational databases [10], while, on the other hand, some of them are based on former techniques such as OMT [5] or even on the E/R model [24]. They therefore have to be updated with UML, SQL:1999 and ODMG 3.0 as their reference models.

This paper is a revised and extended version of Extending UML for Object-Relational Database Design, presented in the UML’2001 conference [17].
In this paper we outline a methodology for object-relational database design. The methodology is based on UML extended with the required stereotypes. We will focus on object-relational databases, although the proposed UML extensions could also be adapted to object database design.

UML [6], as the Unified Modelling Language, is becoming increasingly more accepted. It also has the advantage of being able to model the full system, including the database, in a uniform way. Besides, as UML is an extendable language, it is possible to define the required stereotypes, tagged values and/or constraints, for each specific application. Therefore, in the methodology suggested we propose to use the UML class diagram as the conceptual modelling notation. We also propose some guidelines for translating a conceptual schema (in UML notation) into a logical schema. The logical model used is the SQL:1999 object-relational model so that the guidelines were not dependent on the different implementations of object-relational products. We use Oracle8i as an implementation example. Moreover, comments are also offered here on every issue affected by improvements introduced in the new version, Oracle9i.

Traditionally, methodologies provided graphical techniques to represent a relational schema, such as Data Structured Diagrams (DSD), or some other graphical representation (see, for example, [2]), etc. In the same way, an object-relational schema can be represented either in SQL (SQL:1999 or Oracle8i) or by means of some graphical notation. As the graphical notation to represent the logical schema we also propose to use the UML class diagram extended with the required stereotypes, tagged values and/or constraints.

Both the methodology and the UML extensions have been applied to several cases of study. In this paper, we present as an example a project for the management of reservations for the University’s computer classrooms via the web.

We would like to remark on the importance of providing methodological guidelines for database design using UML for data intensive applications. “Generic lifecycle methodologies for object-oriented development have, to date, not given serious consideration to the need for persistence; either in terms of storage of objects in a relational database or in an objectbase” [7]. Information systems have to manage very many data that need to be persistent. Good persistent data design will improve use and maintenance. Today, the main mechanism of persistence is still the database. As is stated in [12], the IDC market research firm reported global sales revenue for 1999 of $11.1 billion for relational and object-relational databases and $211 million for object databases. Up to 2004, IDC predicts annual growth rates of 18.2 percent for relational and object-relational databases and 12.5 percent for object ones. Therefore, databases will still be the kernel of almost every information system for a long time.

The rest of the paper is organized as follows: Sect. 2 summarizes the current UML extensions for database design. Section 3 proposes new UML extensions for object-relational database design based on the SQL:1999 and Oracle8i object-relational models. Section 4 sums up, by means of an example, the methodology, including some transformation guidelines, using the proposed extensions. Finally, Sect. 5 summarizes the main conclusions and future work.

2 Previous Work

The UML extension mechanism allows for the controlled extension of the language by means of stereotypes, tagged values and constraints [6].

- **Stereotype:** “a stereotype extends the vocabulary of the UML, allowing you to create new kinds of building blocks that are derived from existing ones but that are specific to your problem. This new block has its own special properties (each stereotype may provide its own set of tagged values), semantics (each stereotype may provide its own constraints), and notation (each stereotype may provide its own icon). A stereotype is rendered as a name enclosed by guillemets (for example, << name >>) and placed above the name of another element. As a visual cue, you may define an icon for the stereotype.”

- **Tagged value:** “a tagged value extends the properties of a UML building block, allowing you to create new information in that element’s specification. With stereotypes, you can add new things to the UML; with tagged values, you can add new properties. A tagged value is rendered as a string enclosed by brackets and placed below the name of another element.”

- **Constraint:** “a constraint extends the semantics of a UML building block, allowing you to add new rules or modify existing ones. A constraint specifies conditions that must be held true for the model to be well-formed. A constraint is rendered as a string enclosed by brackets and placed near the associated element.”

For very common applications, such as Web applications, database applications, etc., it would be desirable to provide a standard extension that could be used by every developer. So, Conallen has proposed a UML extension for Web applications [9] and there are also some extensions for database design [1, 6, 19]. Taking into account the previous extensions for database design we propose some specific stereotypes for each of the phases of the development of a relational database (see Table 1).

It will be noticed that Table 1 considers the relational model including stereotypes to represent primary keys, foreign keys, etc. Nevertheless, it does not provide specific stereotypes for object or object-relational models such