Chapter X

Transforming UML Class Diagrams into Relational Data Models

Devang Shah
eXcelon Corporation, USA

Sandra Slaughter
Carnegie Mellon University, USA

ABSTRACT

The Entity-Relationship (ER) method is the most popular method for relational database design. On the other hand, the Unified Modeling Language (UML) is widely used in object-oriented analysis and design. Despite the increasing use of object-oriented techniques for software design and development, there is a large installed base of relational databases. Additionally, object-oriented databases are still not in widespread use. Thus, software designers and developers often turn to the relational databases to make their application objects persistent. Considering the fundamental differences between the two methods, the transformation from UML to a relational data model could be a non-trivial task. The purpose of this chapter is to describe a process that can be used to map a UML class diagram into an ER diagram, and to discuss the potential of using the UML notation to draw ER diagrams. An example of an actual systems design is used throughout to illustrate the mapping process, the associated problems encountered, and how they could be resolved.

INTRODUCTION

The Entity–Relationship (ER) model is the most widely used data model for the conceptual design of relational databases. It focuses solely on data, representing a “data network” that exists for a given system. It has emerged as the leading formal structure...
for conceptual data representation and has become an industry standard. The ER model is based on only a few modeling concepts and has a very effective graphical representation in which each element of the model is mapped to a distinct graphical symbol (Batini, Ceri, & Navathe, 1992).

In the past few years, the Unified Modeling Language (UML) has emerged as a prominent modeling language for object-oriented analysis and design (see Oestereich, 2002; Fowler & Scott, 2000; Booch, Rumbaugh, & Jacobson, 1999; Rumbaugh, Jacobson & Booch, 1999). The class diagram is an important part of the UML, as it captures the static view of the system. The class diagram models classes in the real world and specifies the relationships between them. The underlying concept of class diagrams may seem to be similar to that of ER diagrams; however, there are a few fundamental differences between the two modeling languages. Usually, the ER model is used with the method Structured System Analysis and Design, which is primarily process-centric (see Pressman, 1997; Meilir Page-Jones, 1988; Yourdon, 1988). On the other hand, object modeling is a part of the method (Object Oriented Analysis and Design), which is primarily functional/data-centric (see Muller, 1999; Bahrami, 1999; Dewitz, 1995). Having said that, if we ignore the method/operation property of objects, we can say that object modeling, in concept, is very similar to data modeling. As Rumbaugh et al. (Rumbaugh, Blaha, Premerlani, Eddy & Lorensen, 1991) have observed, the Object Modeling Technique (OMT) is an enhanced form of ER that includes some new concepts (such as qualification). Thus, UML is an enhanced form of OMT and an enhanced form of the ER model (Ou, 1997).

Although object-oriented methods enjoy some success in the software development field, software engineers often turn to ER diagrams and relational databases to implement the objects, i.e., to make them persistent (Muller, 1999). This raises a number of important issues. If the class diagram is a superset of the ER diagram, then why do we need a separate notation to draw the ER diagrams? Can the UML class diagram notation be used to draw the ER diagrams? What is the advantage of that? How would the UML class diagram handle different constructs of the ER diagrams such as primary key constraint, referential integrity constraint, or unique key constraint? What about normalization?

Translating a class diagram into an ER diagram could be a non-trivial task, as several symbols and notations used in the class diagram (e.g., n-ary relationships, aggregation, composition) do not have direct mappings to the ER diagram. A logical and a physical relational database design will require a systematic, step-by-step process to translate a class diagram into the ER diagram. This chapter discusses a process that can be used to simplify the database design task of making an object persistent. It explains how to transform UML class diagrams into relational data models. Some have argued that object modeling is the same as relational modeling. Others have confronted this view; however, we do not delve into that issue in this chapter. Our intent is to examine the efficacy of the UML class diagram as a vehicle to draw the ER diagram. We illustrate data modeling in UML using a real example from an extranet-based retail pharmacy drug dispensing system that was designed for a regional health care network of hospitals, pharmacies, pharmacy brokers, patients, and drug manufacturers. We also discuss some of the challenges faced by application developers in implementing object-oriented relational database applications and several tools and application programming interfaces that address these challenges.
Related Content

Using Device Detection Techniques in M-Learning Scenarios
wwwirma-international.org/chapter/using-device-detection-techniques-learning/73176/

New Dimension in Relational Database Preservation: Using Ontologies
wwwirma-international.org/chapter/new-dimension-relational-database-preservation/73178/

Native XML Programming: Make Your Tags Active
wwwirma-international.org/chapter/native-xml-programming/27781/

Rapid Pattern-Oriented Scenario-Based Testing for Embedded Systems
wwwirma-international.org/chapter/rapid-pattern-oriented-scenario-based/29615/

Introducing Non-functional Requirements in UML
wwwirma-international.org/chapter/introducing-non-functional-requirements.uml/30540/