Implementation of Object-Relational DBMSs in a Relational Database Course

Ming Wang
Department of Computing and Mathematics
Embry-Riddle Aeronautical University
Daytona Beach, FL 32114

E-mail: wangm@db.erau.edu

Abstract: Object-relational DBMS was gradually added as a new topic to the author's database course in response to the rapid changes in DBMS technology in the real world. Implementation of ORDBMS technology in a traditional relational database course had significant impacts on the database curriculum. As an outcome, students were able to solve problems that could not be solved well in a relational database. ORDBMS was implemented with Universal Modeling Language (UML) and the Oracle 8i server. Course design, teaching methodology, class activities and the outcome of the course are discussed.

1. Introduction

Database technology is in a period of intensive change and innovation that is both revolutionary and evolutionary. The revolutionary aspect refers to the innovation of an object-oriented database management system (OODBMS) based on Object-oriented programming technology. The evolutionary aspect refers to the promotion of a new extended version of relational database technology under the name object-relational database management system (ORDBMS).

Object Relational DBMSs involve the extension of relational database systems to add object-oriented features or direct representation of application objects in relational databases. ORDBMS may be the most appropriate choice for a database management system (DBMS) that processes complex data and complex queries according to Stonebraker's four-quadrant view of the database world [1].

According to research published in PC Week Magazine 1999 [2], RDBMS and ORDBMS might dominant the DBMS market for at least the next 10 years. OODBMS will be used for some specific applications. Most companies are not ready to switch to OODBMS because of the heavy investment in RDBMS and unsolved problems in the OODBMS.

Due to the rapid development of database technology, we need to bring ORDBMS to the classroom. The reasons are as follows:

1. Integrating ORDBMS in a relational database course can resolve many of the weaknesses of relational databases not realized by our students. Mastery of ORDBMS technology will allow students to solve more complex database problems in their future careers.

2. ORDBMS will help students better understand object-oriented concepts such as encapsulation and inheritance. The beauty of Object-oriented design is reuse and sharing data. The main advantages of extending the relational data model come from reuse and sharing [3]. If multiple applications or tables use the same set of database objects, then you have created a de facto standard for the database objects.

3. With the OODBMSs still under development, the new Object-Relational emerging technology has been adopted by most database vendors. To overcome the identified weakness of relational DBMS, three of the leading RDBMS vendors, Oracle, Informix, and IBM have all extended their systems to ORDBMS. Learning ORDBMS will help students better prepare for their career development since OODBMS has come into the mainstream of the database field.

4. ISO SQL3 continues its evolution toward adding object-relational features to database management. Even though the official release of SQL3 is behind, SQL3 (ISO 1998a version) already includes many object-relational features [4].

SIGCSE 2001 2/01 Charlotte, NC USA

Based on the above reasons, we need to bring ORDBMS technology into our relational database course even though we already have more than enough topics to cover in our current relational database course. This article was written to present an overview of integrating object-relational DBMSs into a relational database course. The purpose was to help database instructors who have a desire to teach object-relational DBMS in their class. Five topics are described below.

1. Background of ORDBMSs
2. Course design
3. Design and implementation tools
4. Class Activity
5. Outcomes

2. The Background of ORDBMSs

The success of Relational DBMSs in the past decades is evident. However, the basic relational model and earlier version of SQL proved inadequate to support object presentation. It has been said that traditional SQL DBMSs experience difficulty when confronted with the kinds of "complex data" found in application areas such as hardware and software design, science and medicine, document processing, mechanical and electrical engineering, etc.

To meet the above challenges, the object-relational DBMS emerged as a way of enhancing the capabilities of relational DBMS with some of the features that appeared in object-oriented DBMSs. Object-relational DBMSs are supposed to combine the traditional benefits of relational systems with the ability to deal with complex data—a kind of "one size fits all" solution to the database management problem.

The concept for the ORDBMs is a hybrid of the RDBMs and OODBMs. The ORDBMS provides a natural and productive way to maintain a consistent structure in the database. Query and procedural languages and call interfaces in ORDBMSs are familiar: SQL3, vendor procedural languages, and ODBC, JDBC, and proprietary call interfaces are all extensions of RDBMS languages and interfaces. As an evolutionary technology, the object-relational DBMS has inherited the robust transaction- and performance-management features of its relational ancestor and the flexibility of its object-oriented cousin. Database designers can work with familiar tabular structures and data definition languages (DDLs) while assimilating new object-management possibilities.

3. Database Course Design

Topics covered in the entire database system course are as follows:

1. Data Modeling using the Entity-Relationship Model
2. Enhance Entity-Relationship and Object Modeling
3. Relational Data Model and Constraints
4. Relational Algebra
5. Implement SQL in Oracle
6. Mapping ER/EER to Relational Schema
7. Functional Dependencies and Normalization
8. Object-relational database Features
9. Object-oriented Database
10. Oracle PL SQL
11. Oracle JDBC programming: Embedded SQL

Specific topics covered in the object-relational database part are as follows:

1. Object-relational DBMS Design
2. User-defined object types and user-defined methods
3. Type constructors
4. Manipulating composite data types
5. Data inheritance
6. Update-query and Select-query
7. Calling user-defined objects and methods from JDBC

During the semester, ORDBMS concepts were repeatedly introduced at appropriate points. The sequence was as follows:

1. Introducing ORDBMS concept with other data models.
2. Introducing ORDBMS design and UML after relational database design and ER diagram.
3. Converting conceptual ORDBMS database schema from UML to ER diagram.
4. Mapping conceptual ORDBMS design to logical design
5. Implementing ORDBMS such as user-defined data types and methods
6. Calling those defined objects and methods on the server in JDBC programming.

Learning Objectives

The student should be able to

1. describe object-relational DBMSs, and how they differ from relational DBMSs
2. explain the concept of data inheritance in an ORDBMS
3. explain how an ORDBMS can be used to solve problems associated with complex data
4. implement base data type extensions and user-defined functions and operators
5. implement update and query rules

The OODBMS concept was covered very briefly as a new trend in the author's class. The learning curve for using OODBMS is much longer than ORDBMS since ORDBMS is just an extension of RDBMS. It would take a whole course to teach OODBMS and its implementation. It is appropriate to offer OODBMS as an advanced level undergraduate level or graduate level course.

4. Design and Implementation Tools

1. Unified Modeling Language (UML)

UML is used as a tool for ORDBMS design. UML is a new modeling tool developed by the Object Management Group. UML development was spearheaded by Rational Software Corp. Although the UML technology was developed mainly for software design, the important part of this technology, classes and methods, are roughly equivalent to ORDBMS types and methods. In UML class diagrams, a class is displayed as a box (see figure 1) that include three sections: the top section gives the class name; the middle section includes the attributes for individual objects of the class; and the last section includes methods that can be applied to these objects.

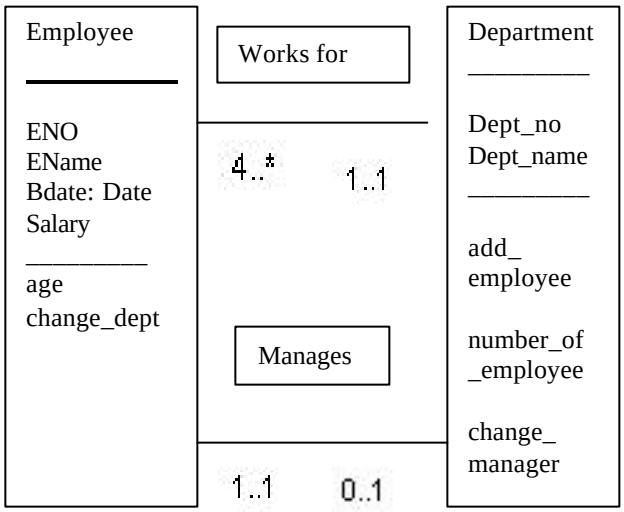

Figure 1. UML Class Diagram [5]

2. Oracle Database Server

The Oracle 8i database server fully supports the Object-relational database model by providing abstract data types, nested tables, varying arrays, binary large objects (BLOB) and character large objects (CLOB) [6]. This provides higher levels of abstraction so that application developer can manipulate persistent objects as opposed to constructing the data from relational data. Moreover, object type declarations can be reused via inheritance, thereby reducing application development time and effort. Three tools inside Oracle 8i were utilized in the author's class as follows:

1. SQLPlus was used to define collection data type.
2. PL/SQL was used the implemented user-defined objects and method.
3. JDBC connectivity was used for application programming.

5. Examples of Class Activities

The class activities were based on solving problems in the real database world. The strategy was to pick a mini-world scenario and let students apply the relational database approach followed by the object relational approach. Then let students draw conclusions by themselves through comparison. The outcome was that students came to recognize that the ORDBMS provides better solutions to the problems they worked on. In this way, students were introduced to applications where an ORDBMS is appropriate.

1. Normalization Problem

In a relational database class, students usually split an address attribute into four fields such as street, city, state and zip code in order to store address information in a customer table. This violates the Third Normalization Form (3NF) indicating transitive dependency in the customer table. To comply with relational database rules, we provide students with the following three solutions:

a. To denormalize the table to the Second Normalization Form (2NF)
b. To create a new customer address table by splitting the address field from the original customer table
c. To store all the customer address information in one field

None of the above is considered as successful. The first solution is not satisfactory because 2NF is not an ideal form in relational database design. The second solution implies that one more join must occur in the query process since one more table has been added to the database. The third solution creates difficulty in data retrieval. For example, it is impossible to retrieve or sort customer records by city, state or zip code.

In the object-relational database class, address can be defined as a user-defined abstract data type with a number of fields if you create standard data types to use for all addresses, then all of the addresses in your database will use the same internal format. Once address is defined as a user-defined data type, it can be re-used within any table

such as customer or employee in a database without concerning violation of 3NF.

2. Speed up query processing

In a relational model, multi-valued attributes are not allowed. One solution to the problem is that each multiple-valued attribute has to be handled by forming a new table. If five fields of a large table were multi-valued, we would have six tables generated from a single table after the First Form of normalization. To retrieve the data back the student would have to do five joins across these six tables. The second solution is to flatten each value in the multi-valued field into a new individual field. The common multi-valued examples in the real world are as follows:

Author, Car-color, and Phone-number

ORDBMs allows multi-valued attributes to be represented in a database. In Oracle 8 either the varying length array (VARRAY) data type or a nested table can be used as a new data storage method for multi-valued attributes.

3. Call stored compiled functions defined on the server in programming

Reuse comes from the ability to store standard functionality on the server, rather than have it coded in each application. For example, applications may require spatial data types that represent points, lines, and polygon, with associated methods that calculate the distance between the two points. If this functionality is embedded in the server, it saves having to define them in each application that needs them, and consequently allows the functionality to be shared by all applications.

4. **Ruse User-defined data objects**

The main advantages of extending the relational data model come from reuse and sharing. If multiple applications or tables use the same set of database objects, then you have created a de facto standard for the database objects. For example, if you create standard data types to use for all addresses, then all of the addresses in your database will use the same internal format.

6. Outcomes

By utilizing Object-relational DBMS students were able to solve the problems that can not be solved well in the relational database. Meanwhile students refreshed the Object-oriented design and object-oriented programming paradigm they learned from the other courses. Students who had experience with C++, Java or Rose Rational often came up with questions about class design in an object-relational database design. They asked if there was any difference in class design between object-relational databases and object-oriented programming. Naturally they tried to tie object-relational databases and object-oriented programming together.

References

[1] Stonebraker M., Object-relational DBMSs: the Next Great Wave. San Francisco, CA: Morgan Kaufmann Publishers, Inc. 1996

[2] Preview, SQL database server, PC Week Magazine Vol. 18, No, 14, 1999

[3] Database systems: A practical approach to design, implementation, and management, 2nd Ed. by connolly, Begg and Strachan, 1998

[4] Sql-3: Implementing the Object-Relational Database, *Fortier, Paul* , Osborne McGraw-Hill, May 1999

[5] Fundamental of database Systems, 3rd Ed. by Elmasri and Navathe, Addision-Wesley 2000

[6] Bringing object-relational technology to the mainstream, Krishnamurthy, Banerjee and Nori, Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data and Symposium on Principles of Database Systems, May 31 - June 3, 1999, Philadelphia, PA