The Online Grade Book—
A Case Study in Learning about Object-Oriented Database Technology

Charles R. Moen, M.S.
University of Houston - Clear Lake
crmoen@juno.com

Morris M. Liaw, Ph.D.
University of Houston - Clear Lake
liaw@cl.uh.edu

Abstract

An object-oriented database is faster than a relational database, and it can store complex data more efficiently. However, object-oriented database technology is not as well known as relational database technology. This paper examines how students at the University of Houston - Clear Lake (UHCL) learn to develop an object-oriented database by completing a semester-long project.

Introduction

Relational databases are routinely used for data storage in modern programs, but object-oriented databases can be a better alternative.

An object-oriented database has many advantages over a relational database. One advantage is that when an application is based on an object-oriented language, it can store that application's data objects without translating them to a different data structure every time they are saved. And vice versa, the stored objects can be loaded into the program without translation. This efficient data storage and retrieval help speed up the application.

A second, more important advantage is that an object-oriented database is the best choice for storing complex data. An important reason for this advantage is that an object-oriented database does not store data in tables. In contrast, a relational database does store its data in tables, and if the data is complex then it must be stored in multiple, related tables. Retrieving that data requires the use of joins, which are very time-consuming operations. An object-oriented database is not slowed down by these limitations. Even if its data does have to be retrieved from multiple objects, then it can be retrieved very quickly by pointer traversal.

Despite their advantages, object-oriented databases are unfamiliar to many database developers.

The Course

At UHCL, students have the opportunity to study object-oriented database technology in CSCI 5433, a graduate-level course called "Object-Oriented Database Systems."

The course has two formal prerequisites: 1) CSCI 3233 Object-Oriented Design and Programming, and 2) CSCI 5333 Database Management Systems. CSCI 3233 is a course in programming with C++, and CSCI 5333 is a course that explores database construction and theory, especially in the context of relational database systems.

The textbook is *Object-Oriented Modeling and Design for Database Applications* (Blaha and Premerlani).

There is one semester-long project, and there are two exams, a midterm and a final.

There are two learning goals for the course. They are: 1) to learn how to use object-oriented analysis and design techniques in the context of developing an object-oriented database, and 2) to learn how to create an object-oriented database and how to develop a Web application that uses it as a backend.

The first goal is achieved by teaching the student how to use the Object Management Technique (OMT) Methodology. This methodology, which was developed in the early 1990s by James Rumbaugh, consists of a process for object-oriented development and a notation for the models that are created in the analysis stage of project development.

The six steps of the OMT process are summarized in Table 1. The students are required to use the first five steps of this process for the semester-long project. They are also required to document these steps in three written reports—the Conceptualization Report, the Analysis Report, and the final Project Report.

Table 1. The OMT process for database development

1. Conceptualization	Conceive of a new application, write the problem statement, list the requirements, and write the use cases
2. Analysis	Create an object model based on entities in the problem statement, and create a functional model based on the use cases
3. System Design	Specify the high-level system architecture
4. Detailed Design	Adjust the models so they can be implemented more easily
5. Implementation	Translate the designs into code
6. Maintenance	The application goes into daily use

The second learning goal is addressed by requiring the students to complete the project assignment, an important part of this course. In the assignment, the students are asked to develop an object-oriented database and a realistic Web application that uses it as a backend. A single project is assigned, and then the class is randomly divided into teams of two or three students. Each team is asked to work independently on their version of the project, and the completed project must be demonstrated in the final class of the semester.

In addition, each team is required to construct and maintain a team home page that has links to their project reports, the project, and a diary of their work.

ObjectStore

The software that the students use for building the database and the Web application is an enterprise-class object-oriented database management system called ObjectStore. ObjectStore is a product of ObjectStore (www.objectstore.net), a division of the Progress Software Corporation (www.progress.com), and it is available to schools through an educational license program. It provides data management for either C++ or Java applications, and it includes many useful tools, such as Blueprint and the Object Managers. Blueprint converts UML class diagrams to either C++ or Java code, and the Object Managers can be used to enable the database to store video, audio, or HTML files as objects.

The version of ObjectStore used in this course requires Microsoft Visual C++, because there are several ObjectStore wizards that work within Visual C++. The

programmer uses these wizards to automatically generate some of the C++ code that is used for the database.

The students in this class develop their projects in a special lab with seven Windows 2000 PCs dedicated to object-oriented database development. ObjectStore is installed on these PCs, as well as Microsoft Visual C++, Internet Explorer, and Microsoft Office.

The Online Grade Book

The project for the fall 2002 semester was the Online Grade Book. The assignment was for each team to create a Web application where students could check their grades in this course by navigating to the application's home page and logging in with their student ID. Other requirements were: 1) the grades have to be recorded by the teacher in an object-oriented database, and 2) the overall course grade for each student has to be calculated automatically according to percentage weights that are assigned to each course component by the teacher.

To illustrate how ObjectStore is used in the development process, this section looks at how the Online Grade Book project was implemented by one of the teams. This team completed a particularly successful version.

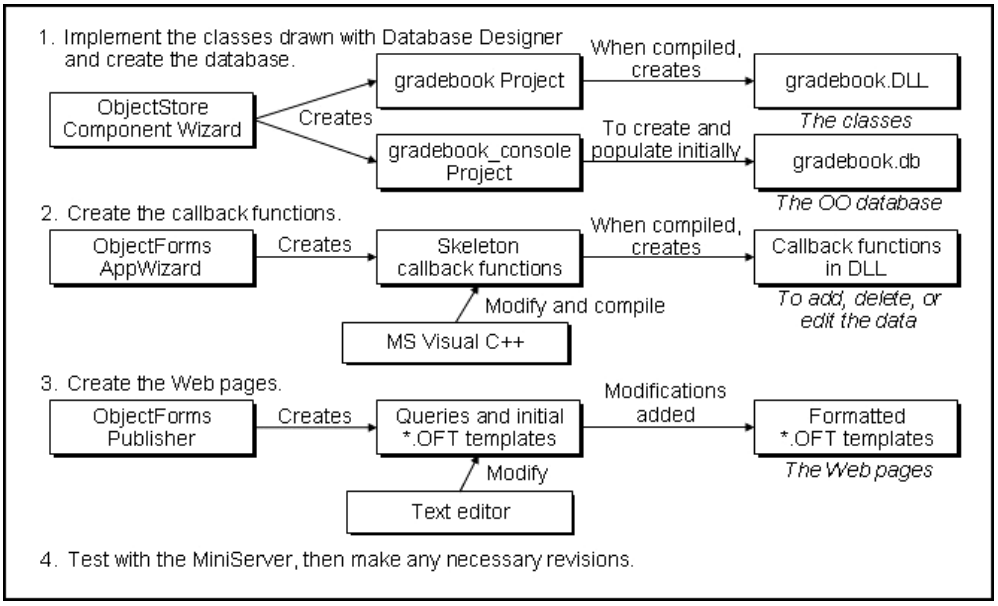

Fig. 1. Tasks in the Implementation Step

In many ways, implementation was the most complicated step in the Grade Book project. The team completed all the implementation tasks shown in Figure 1. There are many tasks and many ObjectStore tools that are utilized in this step, and this article shows highlights of three of the tasks: 1) implementing the database structure by coding the classes in C++, 2) creating the database file, and 3) creating the OFT templates that generate the Web pages.

1. Implementing the database structure

The process of implementing the database structure starts with an object model. This

model is a class diagram that has a class for every type of object to be stored in the database. The team based their model on their study of the problem description and the domain, and they had developed it earlier, in the Analysis Step, with pencil-and-paper sketches. Before they started implementation, they drew a computerized version of it with the ObjectStore Database Designer program. When it was finished, their computerized model had two classes: 1) a Person class, for storing student data, and 2) a Weight class for storing the percentages used to calculate the overall course score for each student. They saved it as "gradebook.dbs."

After completing the computerized model, the team coded the classes by using the ObjectStore Component Wizard in Visual C++. This wizard opened gradebook.dbs and automatically generated the basic C++ code for every class in the object model. For each class, the wizard created both a "cpp" file and a header file. These class definitions included constructors, destructors, and "set" methods for every data member. Figure 2a shows a method from one of the classes.

a.
```
void Person::set_CourseGrade(char *value){
    if(value) {
        int len = strlen(value);
        if(len < 10)
            strcpy(CourseGrade, value);
        else {
            strncpy(CourseGrade, value, 9);
            CourseGrade[9]=0;
        }
    }
    else
        CourseGrade[0]=0;
}
```

b.
```
void Person::set_CourseGrade(double score) {
    if( score >= 93.0 ) {
        strcpy(CourseGrade, "A");
    }
    else if( score >= 90.0 ){
        strcpy(CourseGrade, "A-");
    }
/* ... */
    else {
        strcpy(CourseGrade, "F");
    }
};
```

Fig. 2. Methods in the Person class: *a.* generated by the wizard; *b.* added by the team

The team then added methods to the classes for all the operations that were more than straightforward attribute updates. Since these methods were not routine accessors, they could not be generated automatically. For example, the program had to calculate the overall course score for each student, and it had to calculate the student's letter grade on the basis of the course score. Each of these operations had to be added as a method in the Person class that represented a student. Figure 2b shows a condensed example of how the operation for calculating the student's course grade was implemented by the team as an overloaded form of the "set_CourseGrade" method. In the example, "/*…*/" indicates that code was removed to save space in the figure.

The team then compiled the class definitions to create a DLL that would provide the database structure for the program.

2. Creating the database file

Before any objects could be stored, the team had to create a database file. They started by using the Component Wizard to create a console program that would do this task. The left side of Figure 3 shows the critical code that the wizard generated. The top line of this code will create and open a database file, but it was disabled by the comment symbols. To activate the command, the team only had to remove the slashes and add the name for their file, "gradebook.db." Once these small modifications were made, they ran the console program and it created the database file.

```
//os_database * db = os_database::open("", 0, 0666);
//db->close( );
```

```
os_database * db = os_database::open("gradebook.db", 0, 0666);
db->close( );
```

Fig. 3. Changing the automatically-generated code in the console program

3. Creating the OFT templates

The team also had to create the OFT templates. ObjectStore uses these template files to publish the results of database queries as Web pages. They are text files that are stored on the server, and they contain HTML and special ObjectForms tags. The tags are server-side code that can either invoke a query or help create the HTML that displays the result. They all begin with "<%," and the left side of Figure 4 shows an example.

The team created their initial templates with ObjectForms Publisher, an ObjectStore program that uses a wizard-like tool to create a query, store it in the database, and generate a basic template file to display the query results. Once the team created the initial files with Publisher, they used a text editor to modify them to enhance the Web page appearance.

```
<%include "Headers\GradeBookHeader.html"%>
<%osdatabase="C:\gradebook\gradebook_console\gradebook.db"%>
<%QUERY NAME="query1" OSFUNCTION="osiquery"%>
<%if query1.Status EQ 0%>
<TABLE BORDER=0 CELLPADDING=7 CELLSPACING=0 width="745">
<!-- table title and headers go here -->
<%BEGINDETAIL NAME="query1"%>
      <TR>
      <TD ALIGN="left" BGCOLOR=#FFFFFF><%CourseID%></TD>
      <TD ALIGN="center" BGCOLOR=#FFFFFF><%MidTermWeight%> %</TD>
      <TD ALIGN="center" BGCOLOR=#FFFFFF><%FinalWeight%> %</TD>
      <TD ALIGN="center" BGCOLOR=#FFFFFF><%PresentationWeight%> %</TD>
      <TD ALIGN="center" BGCOLOR=#FFFFFF><%GroupProjectWeight%> %</TD>
      <TD ALIGN="center" BGCOLOR=#FFFFFF><%IndividualProjectWeight%> %</TD>
      </TR>
<%ENDDETAIL%>
</TABLE>
<%else%>
<%query1.ErrorMsg%><p>
<%endif%>
</body></html>
```

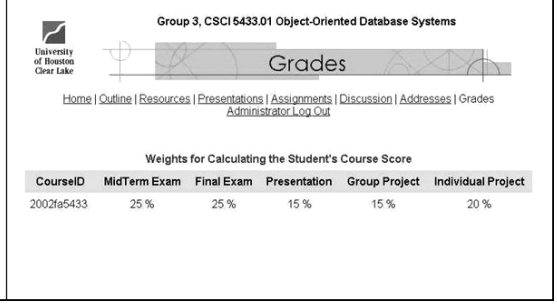

Fig. 4. Left—OFT template for displaying the result of a query as an HTML table. Right—a Web page generated from an OFT template.

There were more tasks that the team had to accomplish, such as creating the C++ callbacks. Even though this article could not describe every step in the project, it shows the essence by describing some of the most important implementation tasks.

Conclusion

UHCL students have been successful in learning object-oriented database technology by using real-world tools like ObjectStore to develop their own realistic application in the semester-long project. Putting knowledge into practice has strengthened their lesson.

Reference

Blaha, Michael, and William Premerlani. *Object-Oriented Modeling and Design for Database Applications*. New Jersey: Prentice Hall, 1998.