CSE 532: Project 2
Music Fan Club
Using Object-Oriented Extensions of SQL

Fall 2009

Deadline: November 11

In this project, you will be designing and implementing part of the Music Fan Club knowledge base (MFCKB) The technical objective of the project is to learn about object-relational extensions of SQL as they are implemented in Oracle.

You will be using Java/JSP/Servlets to build an application front end to your database. Database connectivity should be done using Java/JDBC. The most convenient way to work with servlets is to use Eclipse (`http://www.eclipse.org/`) IDE.

You are required to develop your project using the Subversion (also known as SVN) repository provided by the department and we will check that your code compiles using Eclipse. This is because your project submission will be SVN-based, and TAs will be verifying that everything compiles and runs in Eclipse.

You are allowed to work with a partner—see Section 8.

1 General Description

Please refer to Project 1 for a description of the MFCKB system.

2 System Users

The users of the system are the music fans who subscribe to the MFC services.

3 Required Data

The data items required by your system are the same as in Project 1. However, unlike Project 1, you are to use *object-relational* design and to utilize the object-relational features of Oracle as much as possible. You should determine the relationships among tables, identify the key attributes, etc. In addition, you should associate indices with your tables to speed up processing of the given queries. Finally, you should specify and enforce referential integrity constraints and other CHECK-style constraints on the data.

4 Queries

Every user should have a way to log into his/her account through the Web browser. Once logged in, the user should be able to ask the following queries, which correspond to the first four queries in Project 1.

User queries.

1. Find the friends who share fondness for at least one singer and at least one song of that singer with the user who is asking this query. The querying user must be logged into his/her account.

2. Find all friends of the querying user who like all the singers that are liked by that user. Again, the querying user must be logged in and this is how your application should determine who is asking the query.

3. Find all users who indicated that they like some song, but did not specify that they like its album's singer (note: the same song can be in several albums and sung by several singers; in that case, it is enough to not specify that the user likes one of the singers in one of the albums).

4. Find all users who indicated that they like some song, but did not indicate that they like *any* of its singers.

5 Interface

The interface to the system should be Web-based with the front end using JSP and back-end using servlets. However, nothing fancy is expected. A basic page with links to the required queries would suffice.

6 SVN Repository

The source code of your project must be maintained in the Subversion (SVN) repository provided by the department. Eclipse has a convenient interfaces to SVN. It is **important** to make frequent check-ins to the repository both as a demonstration of professional proficiency and also to document your history of project development. **Projects with unrealistically short histories will be rejected outright.**

More information on these repositories can be found at

`http://www.cs.sunysb.edu/~kifer/Courses/cse532/project/how-to-svn.html`

This page is also linked from the project page.

You will need to request an SVN repository for your project by filling out the appropriate lines in the class wiki on the Blackboard:
`http://blackboard.stonybrook.edu/webapps/lobj-wiki-bb_bb60/wiki/1098-CSE-532-SEC01-92518/course/Home`.

The first batch of repositories will be posted on that same wiki on Thursday, October 15; then on October 19. Requests posted after October 18 will be handled with low priority, and this will negatively affect your project.

7 Documentation and Submission Instructions

All source files written by you should have a header with the following information:

```
/**************************************************************************
                CSE532 -- Project 2

   File name:         XXXXX
   Author(s):         Name 1  (SBU Id)
                      Name 2  (SBU Id)
   Brief description: ZZZZZZZZZZZZZZZZZZZZZZZZZ
                      ZZZZZZZZZZZZZZZZZZZZZZZZZ
**************************************************************************/
```

The source should satisfy all the coding standards (must be well-designed, indented, commented, use reasonable naming schemes, be understandable).

Your deliverables must include a project report with the following items in this order:

- An Entity-Relationship (ER) design and the diagram of the complete project.
- A clear description of the database scheme, including a discussion of your design decisions. Remember that your design should be object-relational.
- Description of **integrity constraints**, including referential integrity and CHECK-constraints.
- All SQL CREATE TABLE/VIEW commands used to build the database. These statements must include all the applicable FOREIGN KEY and CHECK statements.
- For each query listed in Section 4, the report must supply the SQL statement underlying the query.
- All SQL queries must be listed in a *conveniently* readable form. A simple dump of your program's code that includes these queries will **not be accepted** and serious penalty will be applied.
- A brief user guide. Explain how to install and run your program.

To prepare the report, you can use any word processor, such as OpenOffice, MS Word, or LaTex. But **the only acceptable submission formats** for the project report are **PDF** or **plain text**.
The report and the source code must include the following statement at the top:

> **I (*or* We, *if working with a partner*) pledge my (*or* our) honor that all parts of this project were done by me (*or* us) alone and without collaboration with anybody else.**

7.1 Program Code Requirements

Your code must compile and run in Eclipse (3.2.2 or later) when TAs check it out of your repository (the system gives them the access). In particular, the code must be able to access the database that you created on the Oracle server (see `http://www.cs.sunysb.edu/~kifer/Courses/cse532/project.html`). This is how the TAs will be verifying that your project works. If it does not compile and/or run using this method, a significant penalty will be applied and you will have to make arrangements with the TAs to prove that the project found in your repository does indeed work. You can do that during TA's office hours later, but you will not be allowed to make any modifications to your Project 2 repository after the submission deadline.

7.2 Notifying That You Are Done

There is a Blackboard assignment, called Project 2. You should use that assignment to submit your project report (in PDF or TXT formats only) as an attachment.

When submitting the report, you will also see a text box. Please write something like "I am done" so that we'll know that this is not an intermediate version of the project report and that you are really done. If the project does not quite work or does not work at all, explain the problems in that same text box. We will then see if partial credit can still be given for the project.

In addition, if you were working with a partner, indicate that this is a joint project with such-and-such person (Name and Windows user Id). Only one of the partners needs to submit the joint project report, but both partners must fill out the text box.

8 Teaming

You can do Project 2 jointly in groups of at most two. If you choose to work alone, this will *not* reduce the scope of your project.

There are pitfalls in working with a partner whom you do not know well: it can be a frustrating experience to find out that your partner is a free-loader. It also sometimes happens that one of the partners gets involved in a case of academic dishonesty, and this may reflect negatively on you. So, choose your partner carefully!

The division of work between partners must be **vertical**, not horizontal. This means that the partners must collaborate on each part of the project and be on top of what the other partner is doing. Each student must be aware of and understand the techniques used by his/her partner. **Your final document must clearly state who did what part of the job.**

9 Planning Your Work

You really do not have time to waste and should start working right away. I know that quite a number of people decided to take this course unprepared, with gaps in undergraduate material, Java, not having basic skills with IDEs like Eclipse, or version control systems.

Your first step should be to request an SVN repository by filling out a wiki, as explained in Section 6. Once you get your repository, check it out and start maintaining your project in it.

In parallel, begin by producing an Entity-Relationship design and the ER diagram for the admission system. The diagram must then be translated into the object-relational model. This diagram and its object-relational translation should become part of your project report.

The next step is to create your database and populate it with sample data. Then you should write and test the queries. Following that, start designing and implementing the Web interface. Finally, write and test the code needed to connect the interface to your database.

Use the initial test data set, which we provided for Project 1. However, you might need to add more information to ensure that your queries have enough data to work with.