

CSE 532: Project 2

Implementing the Dating Services Support System Using Object-Oriented Extensions of SQL

Spring 2006

Deadline: March 27

In this project, you will design and implement database support for an electronic dating system using the object-oriented extensions of SQL.

You will be using Java/JSP/Servlets to build the 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 NetBeans (<http://www.netbeans.org/>), which is installed in the lab. It has the Tomcat server in it, which is started and shut down automatically when you start/shutdown NetBeans. Another option is to use Eclipse.

1 General Description

The Dating Services Support System (DSSS) is intended to represent information about the clients of the dating service; their preferences; their past and present dates; and queries that are used as part of running the service. The system has two main functions:

1. Enable the clients to find potential “matches” based on preferences and features.
2. Enable analytical querying by the DSSS personnel (presumably in order to work out better algorithms and make the matching process more accurate).

2 System Users

There are two kinds of users: DSSS clients and DSSS personnel. The clients and the personnel have different privileges. The clients can run queries only about themselves or to perform match queries. The personnel will be able to also run analytical queries.

3 Required Data

The data items required by your system roughly fall into these categories:

- *Information about DSSS clients and their preferences.*
- *Information about the DSSS personnel.*
- *Information about the past and present dates of the clients.*

Note that DSSS personnel can also be clients, and the database schema should reflect that fact.

When arranging the data items in tables, you are to use 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.

Every person in the system must have at least the following information:

- Name

- Address
- Login information
- Date of birth

A client, in addition, has

- A set of preferences (the features that the client wants the potential dates to have). For instance, preferences can include sports, hiking, reading, dance, tall, short, slim, etc.
- A set of client's own features. A client's feature can be an interest (e.g., hiking, fishing) or a physical feature (e.g., tall, slim, blond).
- The set of other DSSS clients whom this client was dating in the past. This information should include the time intervals when dating was taking place (for instance, March 7, 2005 to June 22, 2005).
- Information about DSSS clients whom this client is currently dating. This should indicate when dating began.

4 Queries

You are to implement the following queries. Queries 1 – 4 are for DSSS clients and personnel; queries 5 – 8 are for DSSS personnel only. When a user logs in, the system checks the type of the user and presents suitable queries for that type of users. In queries 1 – 4, if the user logged into the current session is a regular client, the Id of that person is used as a parameter to the queries. If the user is a DSSS staff member, then that user will be able to enter the Id of any DSSS client in those queries.

1. List all clients whom a certain person (whose identity is supplied through the GUI) was dating between one date (entered through the GUI) and another.
2. List all clients whom a person identified through the GUI was dating as of certain date (also entered through the GUI) and such that they had at least one common feature with that person.
Here, dating a certain person as of, say, March 7, 2005 (or as of some other day) means that dating started at time prior or equal to March 7, 2005 and has not stopped dating on March 7, 2005 or before.
3. Same as above, but now we want person's dates to satisfy at least N preferences (N is entered through the GUI).
4. Find all potential "perfect matches" for a particular person.

A perfect match for a person, p, is a DSSS client such that the set of all p's preferences is included in the set of that client's features.

This query involves a non-trivial use of negation.

5. For each DSSS client who is over 30 years old, find the average duration of the dates (time between when dating started and the time by which it was over) as of the current date.

6. For each age category of DSSS clients, find the average duration of of the dates. Age categories include people of ages 20-29, 30-39, 40-49, etc.

The query must *not* depend on the number of age categories and it must be a single query (which may use views).

7. Record the fact that a pair of clients started dating.
8. Record the fact that a pair of clients stopped dating.

5 Documentation and Submission Instructions

All the materials must be burned on a CD and submitted at the end of the class on March 27. The CD must have a clearly identifiable label on it, which should list:

1. Authors' names
2. CSE 532, Project 2

Note that the label must be on the CD itself: an unlabeled CD in a labeled case is not acceptable, because such a CD can get misplaced.

Your complete database implementation must be accompanied by a project document. The document should include the following items in the given order:

- The 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 document must supply the SQL statement underlying the query.
- All SQL queries must be listed in a *conveniently* readable form. Please note that a dump of your program's code that contains the requisite queries will **not be accepted** and serious penalty will be applied for poorly formatted queries.
- A brief user guide. Explain how to install and run your program.

In addition, the *source code* must be submitted and placed in a separate directory. The source should satisfy all the coding standards (be well-designed, indented, commented, use reasonable naming schemes, be understandable).

Pay attention to the aesthetics. Poorly formatted or designed works will be penalized. In the project document, every page must have a footer that contains the names of the authors. This is easy to do with any word processor. The acceptable formats for the project document are **text** (e.g., LaTeX) or **.DOC**. Source code must, of course, be plain text.

The document 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.

6 Teaming

You can choose one partner to do the project **jointly**. You can also choose to work alone, but this will not reduce your scope of work.

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 piggy-backing on you. It can also happen that one of the partners gets involved in academic misconduct, and this may reflect on you. So, choose your partner carefully!

The division of work between partners must be **vertical**, not horizontal. This means that the two 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. We will question your code.**

7 Planning Your Work

You have to plan the work on your own. At the end of the project, you will be asked to present a short (15 minute) demo to the TA (there will be sign-up sheets).

Your first step should be to produce 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 the translation should eventually become part of your project documentation.

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 the forms for display and input of the data. Finally, write and test the code needed to produce these forms and connect them to your database queries.

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.