1 Purpose of the project

The primary goal of this project is to implement a single Java application program that will back a new social networking system called FaceSpace. The core of such a system is a database system. The secondary goal is to learn how to work as a member of a team which designs and develops a relatively large, real database application.

You must implement your application program using Java, Oracle, and JDBC. The assignment focuses on the database backing the social network service and not on the user interface. Hence, NO HTML or other graphical user interface should be produced for this project.

2 Specifics

2.1 Data: The FaceSpace database schema and example data

Your FaceSpace database includes the basic information found in a social networking system such as user profiles, friends, groups, messages, etc. You should choose appropriate data types to make up the attributes of each relation that you create to represent miniworld that is described as follows. You are required to define all of the structural and semantic integrity constraints and their modes of evaluation. For both structural and semantic integrity constraints, you must state your assumptions as comments in your database creation script.

Each user in the social network should have a profile that presents their name, email, date of birth, and the time of their last login.

Users can be friends with one another. Friendship will be considered bilateral (if a friendship exists, both users are friends with one another). Once a user attempts to establish a friendship, it should be considered to be pending until it is approved by the other party. Your system must keep track of the state of the friendship (pending or established). Your system should also keep a record of the date on which a friendship was established.

Users can also belong to group. Your system should keep track of a name and description for each group on the social media service, as well as its membership. Further, each group should be defined to have a membership limit that is set when the group is created.

Finally, you should store messages. A message is sent to a single user. You should keep track of a subject, the body text, the sender, and the date sent for each message. You can assume that messages are constrained to be less than 100 characters.

Once you have created a schema and integrity constraints for storing all of this information, you should should generate sample data to insert into your tables. Generate the data to represent at least 100 users, 200 friendships, 10 groups, and 300 messages.

2.2 Functions: A JDBC application to manage FaceSpace

You are expected to do your project in Java interfacing Oracle 11g server using JDBC. You must develop your project to work on `unixs.cis.pitt.edu`. It is your responsibility to submit code that works in this environment. Further, for all tasks, you are expected to check for and properly react to any errors reported by the DBMS (Oracle), and provide appropriate success or failure feedback to the user. Finally, be sure that your application carefully checks the input data from the user.

Attention must be paid in defining transactions appropriately. Specifically, you need to design the SQL transactions appropriately and when necessary, use the concurrency control mechanisms supported by Oracle to make sure that inconsistent states will not occur.

Your application should implement the following functions for managing FaceSpace:

1. **createUser**
   Given a name, email address, and date of birth, add a new user to the system.

2. **initiateFriendship**
   Create a pending friendship from one user to another.

3. **establishFriendship**
   Create a bilateral friendship between two users.

4. **displayFriends**
   Given a user, look up all of that user's established and pending friendships. Print out this information in a nicely formatted way.

5. **createGroup**
   Given a name, description, and membership limit, add a new group to the system.

6. **addToGroup**
   Given a user and a group, add the user to the group as long as that would not violate the group's membership limit.

7. **sendMessageToUser**
   Given a message subject, body, recipient, and sender, create a new message.

8. **displayMessages**
   Given a user, look up all of the messages sent to that user. Your Java program should print out the user's messages in a nicely formatted way.

9. **searchForUser**
   This provides a simple search function for the system. Given a string on which to match any user in the system, any item in this string must be matched against any significant field of a user's profile. That is if the user searches for "xyz abc", the results should be the set of all profiles that match "xyz" union the set of all profiles that matches "abc". The names of all matching users should be printed out in a nicely formatted way.

10. **threeDegrees**
    This task explores the user's social network. Given two users (userA and userB), find a path,

if one exists, between the userA and the userB with at most 3 hop between them. A hop is defined as a friendship between any two users. The path should be printed out in a nicely formatted way.

11. **topMessagers**
    Display the top k users who have sent or received the highest number of messages during the past x months. x and k should be an input parameters to this function.

12. **dropUser**
    Remove a user and all of their information from the system. When a user is removed, the system should then delete the user from the groups he or she was a member of using a trigger. Note that messages require special handling because they are owned by both the sender and the receiver. Therefore, a message is deleted only when both the sender and all receivers are deleted. Attention should be paid handling integrity constraints.

2.3 Driver: Putting it all together

The primary task for this phase is to create a Java driver program to demonstrate the correctness of your social network backend by calling all of the above functions. It may prove quite handy to write this driver as you develop the functions as a way to test them.

Your driver must include a "run demo" function, which automatically generates a sequence of calls to the different functions described in the previous section. Your demo should illustrate that your project works properly and is able to handle the specified special cases and constraints.

3 Project Deadlines and Milestones

- **Teams: (due April 7th)**: Organize yourselves into teams of 2-3 students and send the TA an email specifying: 1) the names of all team members, and 2) the name of a team leader.
- **FaceSpace v0.1: (due April 15th)**: Submit the SQL scripts necessary for creating your database, and populating it with example data. Additionally, submit a Java code, which implements the first three functions listed above, together with a simple driver that allows calling those functions.
- **Final FaceSpace: (due April 27th)**: Submit all the SQL scripts and Java code needed for executing all the FaceSpace functions, as well as your demo for testing those functions.
- **FaceSpace Demo: (due April 28th)**: A presentation of your project followed by Q&A. All team members must participate in demonstrating their project.

4 Grading

The end restult project will be graded on correctness, robustness and readability. Programs that fail to compile or run or connect to the database server earn zero and *no partial points.*

One goal of this project is to help you start to incorporate feedback into a larger project. Hence, the TA will assign you a grade and give feedback on your submission of FaceSpace v0.1. By incorporating the feedback given by the TA into the final submission, you can earn back some of the points lost on FaceSpace v0.1.

Note that points deducted for issues that illustrate a lack of time or thought put into the project (e.g., unhandled database or compilation errors, schemas that fall far short of being able to handle the data needed to represent the mini world mentioned above, or inappropriate assumptions made in defining integrity constraints) cannot be earned back and will negatively affect your overall grade.

Because of this approach, you may find it to your benefit to submit FaceSpace v0.1 early. The sooner you submit, the sooner the TA can begin grading that milestone and provide you with feedback.