CS 1555/2055 Summer 2016 Term Project

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 Milestone 1: The FaceSpace database schema and example data

Due: 11:59PM, July 11, 2016

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 can be sent to either a single user or every user currently in a single group (exclusive or, you do not need to keep track of message sent to a user and also a group). 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 Milestone 2: A JDBC application to manage FaceSpace

Due: 11:59PM, July 24, 2016

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 (e.g., isolation level, locking modes) to make sure that inconsistent states will not occur. Assume that multiple requests for changes to FaceSpace can be made on behalf of multiple different users concurrently.

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 establish 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 so 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. **sendMessageToGroup**
   This should operate similarly to sendMessageToUser only it should send the message to every member currently in the specified group.

9. **displayMessages**
   Given a user, look up all of the messages sent to that user (either directly or via a group that they belong to). Your Java program should print out the user's messages in a nicely formatted way.

10. **displayNewMessages**
    Operates similarly to displayMessages, but only displays messages sent since the user's last login.

11. **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.

12. **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.

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

14. **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 Milestone 3: Bringing it all together

Due: 11:59PM, August 4, 2016

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. You may also wish to reuse your data generation code to dynamically generate a large number of function calls within your driver.

Now this may not seem like alot for a third milestone (especially since it is stated that you may want to do this work alongside milestone 2). The reasoning for this is to allow you to focus on incorporating feedback from the TA regarding milestones 1 and 2 into your project as part of milestone 3. This will be the primary focus of milestone 3.

3 Project submission

Your project will be collected by the TA via the GitHub repository that you have shared. To turn in your code, you must do three things by the deadline:

1. Make a commit to your project repository that represents what should be graded as your group's submission for that milestone. The message for this commit should be "Milestone X submission" where "X" is 1, 2, or 3.
2. Push that commit to the GitHub repository that you have shared with the instructor and TA.
3. Send an email to the instructor and the TA with the title "[CS1555] Project milestone X submission" that includes a link to your repository on GitHub and the Git commit ID for the commit that should be graded. The commit ID is a 40 character string (a SHA-1 hash) and can be found on GitHub or as part of the output of the "git log" command.

Be sure to send this email to the instructor and TA by the deadline. Pushing the code to GitHub is not enough! If you do not email the instructor and TA a commit ID as specified above by the deadline, your submission for that milestone will be considered late and you will receive a 0 for that milestone.

4 Grading

The end result project will be graded on correctness (including transaction usage), robustness (error-checking, i.e., it produces user-friendly and meaningful error messages) 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 each milestone of the assignment. By incorporating feedback given by the TA into later milestones, you can earn back some of the points lost on earlier milestones. For milestone 1 and 2, the TA will assign you a general letter grade assessing the overall quality of your submission. Due to the fact that this grade can be improved through incorporating feedback, treat this letter grade as more of an indicator of how much work you need to do to improve this portion of the term project than what you have earned for that portion of the project.

Note that points deducted from milestones 1 and 2 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 milestones 1 and 2 early. The sooner you notify the TA that you have completed a milestone, the sooner he can begin grading that milestone and provide you with feedback.

Be careful to ensure that you have addressed all the requirements of a milestone when sending a submission email. Once the TA receives a submission email, the commit ID contained within that email is the one that will be graded for that milestone (i.e., your group cannot send 2 submission emails for the same milestone). Your submission for milestone 3 will be considered your final submission for the term project. DO NOT send a submission email to the TA for milestone 3 until you are completely done with your project and are satisfied that you have addressed all feedback from the TA.