INFS 614 - Database Concepts

Fall 2013

Instructor: Dr. Jessica Lin

Project Assignment

General. Your project is to design and implement a database for an online movie rental company like Netflix (for this project, we will focus on the streaming-only plans). At the back-end, the database manages all of its data in the Oracle database management system. At the front-end, you are required to implement a command-line interface to its users (the database users, not Netflix users) using Java and JDBC.

The following is the list of "basic" information that the database should store. Note the organization of the list does not necessarily imply that this is how you should design the database. For example, for each member, you have a list of items that need to be stored. It is up to you to decide how you should store and organize these items. You will need to come up with a design that makes sense, given the descriptions. *You may work in a team of 2.*

1. Accounts. For each account, record the account owner's information as follows:
    a. Unique member ID
    b. Full name (first name, last name)
    c. Credit card on file
    d. Profiles (see below)

2. Movies. For each movie:
    a. Unique movie ID
    b. Movie name
    c. Year it was produced
    d. Cast
    e. Producer
    f. Genre(s)
    g. Average member rating

3. Actor/Actress. For each actor/actress:
    a. Unique ID
    b. Name (first, last)
    c. Movies that he or she was in

4. Each account can have up to 5 profiles for different members in the household[1]. For example, an account owner "bradpitt" may have a profile for himself "Brad", another profile "Angie" for his wife, and a separate profile "Kids" for his kids. For each profile:
    a. Profile name (not unique across all members, but unique within an account)
    b. Preferred movie genre(s)
    c. Movie Queue (see below)
    d. Rental history (see below)

5. There is a separate rental history for each "profile" created by a member (account owner). Remember a "profile" does not have a globally unique identifier.
    a. Movies rented
    b. Ratings posted by this profile (see below). Rating scale is from 1 to 5, 1 being "Do not like" and 5 being "Like"

6. Each profile has its own movie queue. In the queue, record the movies that he/she wants to see sometime in the future. For each queue,
    a. Movies (movie ID)
    b. The profile owner (i.e. the account owner) manually ranks the movies in the queue by assigning priority numbers for the movies. The priority numbers are consecutive numbers between 1 and *n*, where *n* is the number of movies in the queue. Movies with smaller priority number will be placed first.

The descriptions above summarize the minimal information your database should contain. As mentioned, it does not necessarily imply the organization of the relations. You may re-arrange the attributes or add more attributes. The design decision is yours, but your design should be reasonably efficient, and well justified. You will be graded on both correctness and quality of the design.

Your interface should have minimal functionality including a variety of queries and browsing capabilities over the movies (e.g. the database user should be able to search by genre, movie name, movie star, etc.), profiles, rental history, etc. The user should be able to update the database by inserting or deleting tuples.

You'll be graded for each phase. The project accounts for 25% of your total grade.

[1] This is a new feature on Netflix. Separate profiles and rental history allows more accurate recommendation to its members.

Phase 1: Conceptual Design. (5%) You need to submit an ER diagram for your database according to the project description and the assumption you make. (See HW1 assignment for an example).

Phase 2: Schema Design. (5%) After your conceptual design is finished, you need to translate the ER diagram into relation schema. Submit a copy of your relation schema, the SQL script that you use to implement your database, and insert enough tuples in **each** table, e.g. at least 5 accounts (with at least 2 profiles in each account), 10 movies, 20 actors/actresses, etc.

Phase 3: JDBC Implementation. (12%) Write a program in JDBC that will allow users to access and query your database. Command-line menu is acceptable. More information on this phase will be given later in the semester. In-person demos will be scheduled during the last week of class.

Putting it together. (3%) Prepare a final report. Your report should contain the following:

- Materials from all phases
- README describing how to run your program
- Screenshots on program/query execution
- Write-up: Discuss and justify your design decisions and challenges you encountered. List any part(s), if any, that do not work properly or you did not implement.

Timeline:

September 13: project released

October 1: Phase 1 (ER diagram) due

October 15 (online submission): Phase 2 (schema) due

December 3: Demo and final report due

December 10: Final

Information for Phase 3:

To create and populate your database, you have two options:

1. Run your SQL script from Phase 2. Make sure that in the beginning of the script.

Or

2. Copy & paste your CREATE TABLE and INSERT commands from Phase 2 to your JDBC program. You can also read the script file and parse each command.

Either way, **delete all tables that you are about to create**.

Instead of having command-line menu, you can create a GUI or a web app for extra credit (2%).

Your program should allow the user to do the following. Note, the "user" here means the DBMS user, not the website user.

- View table content: Give user a list of existing tables so that he/she can select one to view the tuples
- Add records: Enter information for new members, movies, actors, etc. Update/delete information.
- Search database: Allow users to search for movies based on title or actor; display all matching movies (movie name, year, and average rating). You should use partial matching instead of exact matching for strings.
- Show rental history for a given profile (you'll need to ask for account info first, and then profile).
- Show rental queue for a given profile.
- Exit the program
- (Extra credit 2: 2%) Open queries – allow user to enter any SQL query and display the query result.

Here is a suggestion on what your (command-line) menu should look like:

When the program starts, prompt the user for his/her login and password. Once the program connects to the database successfully, then display the following menu options. After each selection is executed, your program should go back to the main menu.

1. View table content – list your tables for the user to choose, then upon user selection, display the content (tuples) in the table
2. Insert new record into… (the exact attributes depend on your database design) The design is up to you – you could follow the attribute-by-attribute steps as described below once the user selects a table to insert into, or you could list all the attributes and accepts a string containing all attribute values (delimited by commas).

a. Account
    i. Member ID
    ii. First name
    iii. Last name
    iv. Etc.
b. Movie
    i. Movie ID
    ii. Movie name
    iii. Etc.
c. Actor
    i. Actor ID
    ii. First name
    iii. Last name
    iv. Etc.
d. Profile
    i. Member ID
    ii. Profile name
e. Queue
    i. Member ID
    ii. Profile name
    iii. Movie ID
    iv. Etc.
f. History
    i. Member ID
    ii. Profile name
    iii. Movie ID
    iv. Etc.

There might be more tables depending on your design.

3. Update record
    a. Update (then ask for information to update)
    b. Delete (then ask for record to delete)
4. Search for movies
    a. Movie name
    b. Actor name
5. Show information for a member's profile.
    a. Prompt for member ID and then profile name
    b. See rental history
    c. See rental queue
6. (Extra credit 2) Open queries – allow user to enter any SQL query and display the query result. You will need to parse the query to determine how to execute the query and print the output.
7. Exit