University of Ottawa
School of Electrical Engineering and Computer Science
CSI2132-2016
Database Project Specification: Movie Recommender System

This document contains the requirements as created in the two classes. Note that I made *some minor modification* to the design, in order to limit the scope of this project. (For example, we do not consider TV Shows in this database.)

Instructions
1. Complete this project in a group of two (2) to three (3) students.
2. Demonstrate the project on **Monday April 11th, 2016** in a 15 minute timeslot, as allocated by the TA.
3. Use PostgreSQL to complete this project, together with a language such as Java and JSP, or PhP, to create your Web-based front-end. You are also welcome to create a mobile front end, for use on say Android.

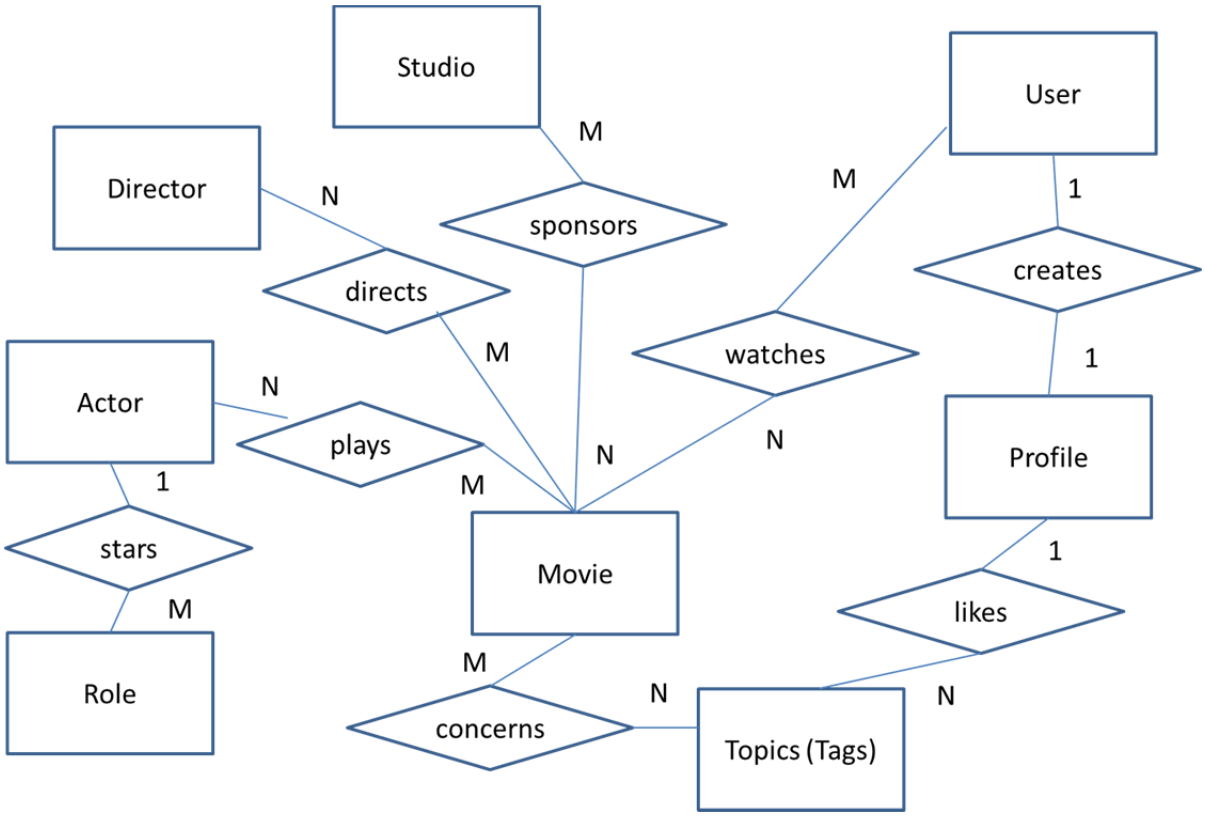
Deliverables:
Submit all your source code via BlackBoard Learn**, before Monday April 11th, 2016 at 13h55**. (That is, before the demonstrations will start.) Note that all group members should submit the source code, not just one per group. All group members should attend the project demonstrations.

Your task:
Consider a database that is to be used to recommend movies to users. In this database, users need to register, select movies to watch, and also rate movies. This database is accessible via the Internet and thus provides a forum for the public to use when deciding what to watch.

(You may want to take a look at http://www.cnet.com/news/top-10-movie-recommendation-engines/ that contains links to recommender systems.)

Here is a list of some assumptions that were made.
1. A user creates a unique profile and this profile is associated with one person only.
2. A user may watch many movies (and vice versa) and may or may not rate the movie.
3. We keep a list of topics (tags such as horror, comedy, family, nature) which a user may use when rating a movie.
4. Similarly, these tags may be used by the website in order to classify movies.
5. An actor may play many roles, but a role is played by only one actor.
6. An actor plays in many movies, and there are many actors in a movie. (The same holds for directors, studios and movies.)

Here is a high level summary of the EER diagram, which only shows the entities and relationships.



Here is a partial definition of the relations you will need to create. You may want to add additional information/attributes to personalise the project. ***You are also welcome to make changes to this design, should you prefer to change the cardinalities and/or connectivities of the relationships as described above.***

- User(UserID, password, last-name, first-name, email, city, province, country…)
- Profile(UserID, age-range, year-born, gender, occupation, device-used, …)
- Topics(TopicID, description) *// e.g. (100, Horror) or (101, Comedy) This table implicitly refer to the genre(s) of a movie.*
- Movie(MovieID, name, date-released, …)
- Watches(UserID, MovieID, date, rating, …)
- MovieTopics(TopicID, MovieID, language, subtitles (y/n), country, …)
- Actor(ActorID, last-name, first-name, DateofBirth, …)
- Role(RoleID, name, …, ActorID)
- ActorPlays(MovieID, ActorID, …)
- Director(DirectorID, lastname, firstname, country, …)
- Directs(DirectorID, MovieID, …)
- Studio(StudioID, name, country, …)
- Sponsors(StudioID, MovieID, …)

Requirements and Mark Allocation

You are required to complete the following tasks: (**Total 100 marks**)

1. **(10 marks)** Transform the description into a relational model and create all the tables in PostgreSQL. Add all other relevant attributes and remember to enforce entity and referential integrity.

2. **(10 marks)** Populate the tables with your own data, using the movies of your own choice. It follows that your data, and the attribute values you choose, should be sufficient in order to implement and test the database. Your database should include at least 40 different movies. There should be at least 20 topics. The number of users should be at least 20, with the highest number of rating by a specific user around 10. It follows that the other tables should be similar in size.

3. **(10 marks)** Provide the user with the ability to add data to, and delete data from, the following tables in your database: Directs, User and MovieTopics.

4. **(40 marks)** Create a number of SQL queries to explore this data. The following is a suggested list of "typical" queries that should be implemented. The general idea is that you should be able to explore the data as contained in your database, in an "ad hoc" fashion.

   <u>Movies, Actors, Directors, Studios and Topics</u>
   a. Display all the information about a user-specified movie. That is, the user should select the name of the movie from a list, and the information as contained in the movie table should then be displayed on the screen.
   b. Display the full list of actors, and their roles, of a specific movie. That is, the user should select the name of the movie from a list, and all the details of the actors, together with their roles, should be displayed on the screen.
   c. For each user-specified category of movie, list the details of the director(s) and studio(s), together with the date that the movie has been released. The user should be able to select the category (e.g. Horror or Nature) from a list.
   d. Display the information about the actor that appeared the most often in the movies, as contained in your database. Display this information together with the details of the director(s) and the studio(s) that s(he) worked with.
   e. Display the information about the two actors that appeared the most often together in the movies, as contained in your database.

   <u>Ratings of movies</u>
   f. Find the names of the ten movies with the highest overall ratings in your database.
   g. Find the movie(s) with the highest overall rating in your database. Display all the movie details, together with the topics (tags) associated with it.
   h. Find the total number of rating for each movie, for each user. That is, the data should be grouped by the movie, the specific users and the numeric ratings they have received.
   i. Display the details of the movies that have not been rated since January 2016.
   j. Find the names, release dates and the names of the directors of the movies that obtained rating that is lower than any rating given by user X. Order your results by the dates of the ratings. (Here, X refers to any user of your choice.)

k. List the details of the Type Y movie that obtained the highest rating. Display the movie name together with the name(s) of the rater(s) who gave these ratings. (Here, Type Y refers to any movie type of your choice, e.g. Horror or Romance.)

l. Provide a query to determine whether Type Y movies are "more popular" than other movies. (Here, Type Y refers to any movie type of your choice, e.g. Nature.) *Yes, this query is open to your own interpretation!*

Users and their ratings

m. Find the names, join-date and profiling information (age-range, gender, and so on) of the users that give the highest overall ratings. Display this information together with the names of the movies and the dates the ratings were done.

n. Find the names, join-date and profiling information (age-range, gender, and so on) of the users that rated a specific movie (say movie Z) the most frequently. Display this information together with their comments, if any. (Here movie Z refers to a movie of your own choice, e.g. The Hundred Foot Journey).

o. Find the names and emails of all users who gave ratings that are lower than that of a rater with a name called John Smith. (Note that there may be more than one rater with this name).

p. Find the names and emails of the users that provide the most diverse ratings within a specific genre. Display this information together with the movie names and the ratings. For example, Jane Doe may have rated terminator 1 as a 1, Terminator 2 as a 10 and Terminator 3 as a 3. Clearly, she changes her mind quite often!

Please note that, in the above, the queries may also return only one name. (That is, there may be only one person satisfying these queries. Be sure to have enough test cases to ensure your queries are robust.)

5. **(30 marks)** Create a web-based front-end, for the user to directly query the database. You may also choose to create a mobile application.

6. Additional effort, such as creating a superb front-end, conducting sentiment analysis, or including a multimedia component, may earn you up to **20 bonus** marks.