Project 1.3 COSI 127B

Due: Thursday 21st March, 2024 at 23:59

In the first project, you will design a simple website like IMDB that uses a movie database. This application will have a simple UI that is connected to a MySQL database.

Quick recap of the first two deliverables:

In the first deliverable, you were asked to create an ER Diagram and Relational schema for an application with the following specifications:

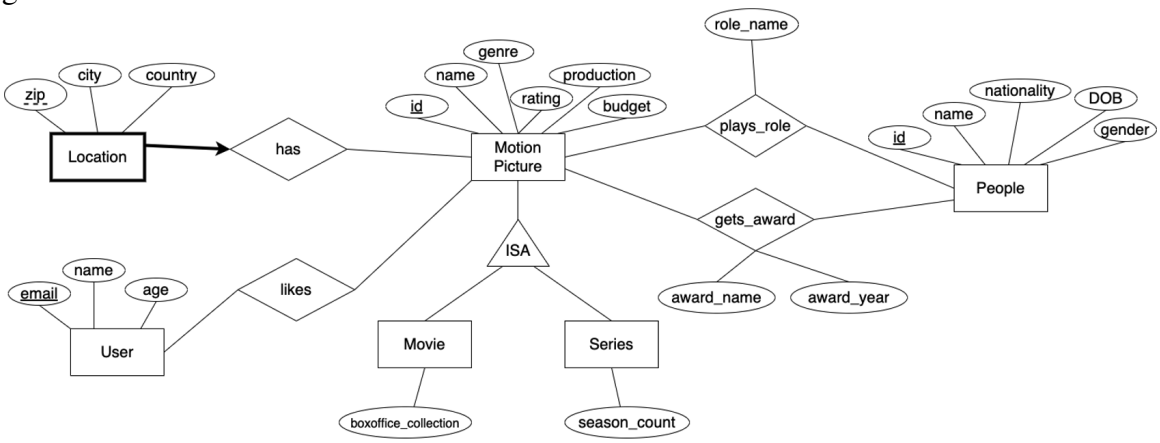
An infotainment startup plans to create a website that will allow its users to view details of various motion pictures. Registered users will be able to view the list of all movies, search movies by name, genre, actor etc. Besides users will be able to 'like' movies. The detailed requirements are listed below:

1. Each motion picture is identified by an id. The database must also store the motion picture's name, genre, production, release date, budget, and ratings. A motion picture can belong to multiple genres, and each genre has a genre id and description of the genre. Ratings are floating values between 0 and 10.
2. Motion pictures can be both Movies and TV series. For TV series, they need to store the number of seasons; for movies, they need to store the box office collection.
3. The database should also contain information about people (id, name, nationality, date of birth, gender) who play a role in any motion picture. Roles can be actor, director, producer, screenwriter etc. Keep in mind that a person can have multiple roles in the same movie.
4. The database should also keep the information of the awards (award name and year received) each person has received for a motion picture. Keep in mind that they are only interested about the individual awards that are specific to people (best actor, best director, best supporting role, etc.) for a motion picture. You do not have to consider the awards that are only specific for motion pictures (e.g., best movie, best cinematography, etc.).
5. To provide better trivia for the users, they need to store the shooting location (name, city, country) for each motion picture. Keep in mind that a movie can have multiple shooting locations.
6. The database needs to keep basic information about the registered users (id, email, name, and age) of the website.
7. Registered users can 'like' motion pictures.

For the second deliverable, we were given a specific schema description and had to convert it to relational tables in MySQL. Further, we had to design a user interface and connect it to the database to support a few basic queries. Thus, here is our existing database setup:

Existing Database Setup

An infotainment startup plans to create a website that will allow its users to view details of various motion pictures. Registered users will be able to view the list of all movies, search movies by name, genre, actor etc. Besides users will be able to 'like' movies. The required ER Diagram for the database is as follows:



Below, we list the tables required in the relational schema along with their primary and foreign keys:

MotionPicture (id, name, rating, production, budget)
User (email, name, age)
Likes (uemail, mpid)
Movie (mpid, boxoffice_collection)
Series (mpid, season_count)
People (id, name, nationality, dob, gender)
Role (mpid, pid, role_name)
Award (mpid, pid, award_name, award_year)
Genre (mpid, genre_name)
Location (mpid, zip, city, country)

Note: Primary keys are underlined and foreign keys are in blue.

Creating a UI

We need a simple UI to execute and display results to the users. Below, we list the basic requirements of the UI:

(a) The user, through the UI should be able to execute a list of queries (we will list the required queries in the next deliverable) that should be parametrized when necessary. For this part (Project 1.2), just for sanity checking, implement few basic queries like – (i) keep a button 'view all movies'; upon clicking it, it will list out all movies and their details, (ii) keep a button 'view all actors', which will work accordingly.

(b) The queries can be executed from the front-end by clicking on links or buttons. Textbox, check boxes or option buttons can be used for parametrization requirements of queries. **Note: only providing a textbox where the user will manually enter every query is unacceptable.**

(c) The results of the queries should be displayed in a tabular format if the result has multiple tuples and columns.

(d) For the requirement of "users liking movies", the UI should support a way of requesting the user's info and accept the user liking a movie through a button. Remember, the likes table must be updated if a user likes a movie.

These are a few basic requirements for the UI, and improvements to the front-end design to meet the required functionality is left to the student's perspective. Please visit the office hours if you need any help.

For the last deliverable, you will modify your application to support a list of queries that would be executed from the front-end onto the database. The results of these queries are to be displayed on the UI.

We again recommend that you start as early as possible on this project.

1 Data Loading

Please download the PA1_3_data.zip file from the Resources tab on Piazza. You will find several csv files that correspond to every table in our schema. You are required to load this data onto each of the table to perform the required queries.
To do so, you can simply select your table in phpMyAdmin console, go to the import tab and load the csv file. A video has been posted on piazza under the resources tab that demonstrates process.

2 Queries

1. List all the tables in the database.
2. Search Motion Picture by Motion picture name (parameterized). List the movie name, rating, production and budget.
3. Find the movies that have been liked by a specific user's email (parameterized). List the movie name, rating, production and budget.
4. Search motion pictures by their shooting location country (parameterized). List only the motion picture names without any duplicates.
5. List all directors who have directed TV series shot in a specific zip code (parameterized). List the director name and TV series name only without duplicates.
6. Find the people who have received more than "k" (parameterized) awards for a single motion picture in the same year. List the person name, motion picture name, award year and award count.
7. Find the youngest and oldest actors to win at least one award. List the actor names and their age (at the time they received the award). The age should be computed from the person's date of birth to the award winning year only. In case of a tie, list all of them.

8. Find the American Producers who had a box office collection of more than or equal to "X" (parameterized) with a budget less than or equal to "Y" (parameterized). List the producer name, movie name, box office collection and budget.
9. List the people who have played multiple roles in a motion picture where the rating is more than "X" (parameterized). List the person's name, motion picture name and count of number of roles for that particular motion picture.
10. Find the top 2 rates thriller movies (genre is thriller) that were shot exclusively in Boston. This means that the movie cannot have any other shooting location. List the movie names and their ratings.
11. Find all the movies with more than "X" (parameterized) likes by users of age less than "Y" (parameterized). List the movie names and the number of likes by those age-group users.
12. Find the actors who have played a role in both "Marvel" and "Warner Bros" productions. List the actor names and the corresponding motion picture names.
13. Find the motion pictures that have a higher rating than the average rating of all comedy (genre) motion pictures. Show the names and ratings in descending order of ratings.
14. Find the top 5 movies with the highest number of people playing a role in that movie. Show the movie name, people count and role count for the movies.
15. Find actors who share the same birthday. List the actor names (actor 1, actor 2) and their common birthday.

The baseline requirement for all parameterized queries is that they at least carry text boxes for us to enter in the parameters required to execute the query. Improvements to the front-end design to meet the required functionality is left to the student's perspective. Please visit the office hours if you need any help.

Additionally, for those who have not completed the "likes" functionality from PA1.2, this is also due with PA1.3.

3 Logistics

3.1 Collobaration

This is a be a group project with maximum size of 2.

3.2 Submitting your assignment

You will need to submit your entire code base and the sql dump file of your local database for the submission. You have two options to submit your assignment:

(a) Upload your code base to a github repo and upload the link to gradescope. Make sure.

(b) Place all relevant files into a single folder, zip it and submit the zipped file to gradescope. (Make sure you discard all irrelevant files (like .Docstore) before clicking on upload.)

3.3 Helper Links

https://getbootstrap.com/docs/4.0/components/forms/

https://www.w3schools.com/