Team member: Jing Wang, Brian Hong

Project title: Movie Rating System

PROJECT URL: <http://flip3.engr.oregonstate.edu:6924/>

**Overview**

In 2019, at least 900 movies were made, and in the last 10 years, approximately 12,000 movies have been released in the United States. According to the U.S Bureau of Labor Statistics in 2019, 52,260 people were employed as actors playing on stage, television, raido, video, films, or other forms of entertainment. Of those, there are approximately 15-30 A-List actors, male and female each, who will make 1-2 movies a year. On average a film director will have made 3 films in their career. These numbers accumulate to a growing list that one person is unable to keep track of by themselves. It can be an arduous task to differentiate which movies are worth their time and money. Fortunately, a website with a database back end can assist in finding out what movies a person’s favorite actor or director is associated with, or how well a movie performed can easily be found.

**Database Outline**

**Reviewers**: records the details of Reviewers. The table contains unique id, name, email of reviewers and id of movies they reviewed.

* *reviewerId*: int(11), auto\_increment,unique, not NULL,**PK**
* *name*: varchar(50), not NULL
* *username*: varchar(50), not NULL
* *email*: varchar(100), not NULL

·   *password:* varchar(100),

* Relationship: a 1:M between **Reviewers** and **Ratings** is implemented with *reviewerId* as a FK inside of **Ratings**.

**Movies**: records the details of Movies. The table contains unique id, title, release year, ratings(scores), genre, budget and box office(revenue) of movies and unique id of directors and actors.

* *title*: varchar(100) NOT NULL
* *movieId*: int(11), auto\_increment,unique,not NULL,**PK**
* *budget*: bigint(20) DEFAULT NULL
* *avgRatin*g: decimal(2,1) DEFAULT NULL
* *genre*:varchar(50), not NULL
* *boxOffice*: bigint(20) DEFAULT NULL
* *year*: int(11), not NULL
* Relationship: a 1:M between **Movies** and **Ratings** is implemented with *movieId* as a FK inside of **Ratings**.
* Relationship: a M:M between **Directors** and **Movies.** A movie can have more than one director and a Director can work on multiple movies.
* Relationship: a M:M between **Actors** and **Movies**. An actor has worked with many directors in the past and directors will have worked with many actors.

**Ratings**: record the details of Ratings. The table contains unique ids of reviewers and movies, and ratings and rating date of a movie.

* *movieId*: int(11),unique,not NULL,auto increment, **FK**
* *reviewerId*: int(11), unique, not NULL,**FK**
* ratingDate: timestamp not NULL default current\_timestamp on update current\_timestamp
* rating:decimal(2,1) default NULL
* review: varchar(255) default NULL
* Relationship: a M:1 between **Ratings** and **Reviewers**.
* Relationship: a M:1 between **Ratings** and **Movies**.

**Directors**:record the details of Directors. The table contains unique ids and names of the directors, and title and unique id of the movie.

* *directorId:*int, auto\_increment,unique,not NULL,**PK**
* *lastName*:varchar(50), not NULL
* *firstName*:varchar(50), not NULL
* Relationship: a M:M between **Directors** and **Movies.** A movie can have more than one director and a Director can work on multiple movies.
* Relationship: a M:M between **Actors** and **Directors**. An actor has worked with many directors in the past and directors will have worked with many actors.

**DirMovies:**

* *directorId:*int, auto\_increment,unique,not NULL,**FK**
* *movieId*: int(11),unique,not NULL,auto\_increment, **FK**
* Relationship: a M:M between **Directors** and **Movies.** A movie can have more than one director and a Director can work on multiple movies.

**Actors**: record the details of actors. The table contains names and unique ids of the actors and id of the movies he/she filmed.

* *actorId*:int, auto\_increment,unique,not NULL,**PK**
* *lastName*:varchar(100), not NULL
* *firstName*:varchar(100), not NULL
* Relationship: a M:M between **Movies** and **Actors**. A movie can have many actors and an actor can appear in many movies.
* Relationship: a M:M between **Actors** and **Directors**. An actor has worked with many directors in the past and directors will have worked with many actors.

**ActMovies:**

* *actorId*:int, auto\_increment,unique,not NULL,**FK**
* *movieId*: int(11),unique,not NULL,auto increment, **FK**
* Relationship: a M:M between **Movies** and **Actors**. A movie can have many actors and an actor can appear in many movies.

**DirActors:**

* *directorId:*int, auto\_increment,unique,not NULL,**FK**
* *actorId*:int, auto\_increment,unique,not NULL,**FK**
* Relationship: a M:M between **Actors** and **Directors**. An actor has worked with many directors in the past and directors will have worked with many actors.

**c) Entity-Relationship Diagram:**

**Diagram

Description automatically generated**

**d) Schema:**

**Diagram

Description automatically generated**

**Feedback by the peer reviewer**

**1)** Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

Comment 1: The Entity names are consistent. For the attributes, 'boxoffice' might need to be boxOffice to stick with the camel case. "ratings" shouldn't be plural. I think it would be better to combine Casts and Directors in to one table, maybe Talents or something. I don't think it makes sense for them to be stored separately. Entity names are capitalized properly.

Comment 2: The naming of entities and attributes is consistent. All entities are plural and capitalized. The attribute "ratings" is plural and should be made singular. The attribute "boxoffice" missed the camel case format. The remaining attributes are all singular and follow the camel case format.

**3)** - Does the outline of entity details describe the purpose of each, list attribute datatypes and constraints and describe relationships between entities?

Comment 1: In general yes, though since "casts" are typically composed of more than one individual, it may be better to add an "actors" entity with a M:M relationship to "casts".

Comment 2: Yes. The purpose of the entities is clear and the relationships between them are all listed. Data types and constraints are listed for all entity attributes. I think the "Casts" entity would be better named as "Actors" because they are using it to represent a single person instead of a group of actors. I also think it might be a good idea to allow for movies to have no actors in the case of a nature documentary or something.

Comment 3: Yes and no. The table on page 1 clearly states which entities each group member is responsible for, but no linking tables were mentioned for any of their three many to many relationships.

**5)** Are 1:M relationships correctly formulated? Is there at least  one M:M relationship?

Comment 1: The 1:M relationships described in the outline all make logical sense. The arrows in the schema for the one to many relationships are all drawn backwards when compared to the lecture though. There are three M:M relationships, but none of them are represented by linking tables.

**Actions based on the feedback**

Based on the comments, we changed boxoffice to boxOffice, ratings to rating, and changed the name of entity Casts to Actors.

We also added linking tables and this change reflects in the ER diagram below.

**Upgrades to the Draft version**

No other changes were made other than listed above.

**a) Fixes based on Feedback from Step 1:**

In our original draft, we were missing a second M:M relationship. In this updated version, we identified the two more M:M relationships. One between Casts and Directors who have worked with one another in the past. Another between Movies with more than one Director, and Directors who have worked on many Movies. We also noticed an error where our Foreign Keys were being auto-incremented, but should not be, as they are referencing the Primary Keys of other entities. We have corrected this mistake.

**b) Project Outline and Database Outline - Updated Version:**

**Entities that each group member will be responsible for:**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Entity** | **Relationship** |
| Jing | Reviewers | 1:M - Ratings |
| Jing | Movies | 1:M - Ratings  M:M - Directors M:M - Actors |
| Brian | Ratings | 1:M - Reviews 1:M - Movies |
| Brian | Directors | M:M - Movies M:M - Actors |
| Brian | Actors | M:M - Movies M:M - Directors |

**a) Feedback by the peer reviewer:**

**Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.**

* Yes, each table in the schema is represented by a list with a hyperlink. If you want to view the movies you would go to the specific Read/Display page.
* Yes - Brian & Jing's website UI has a SELECT for each of their five tables (i.e. Reviews, Movies, Ratings, Directors, and Actors). Currently in the main page they have a list of the tables, and under each table there is a link for SELECT, which leads to a separate page that displays the table entries.
* Yes, every entity has a read page that shows every attribute except id.
* Yes, I can see options to select/read/search every table

**Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?**

* Movies has a search feature page, it is not apparent whether the search page will have a dynamically populated list of movies. I believe it would redirect you to the movie read/display page with your search result, or it may generate it on the page I'm not sure.
* Yes - In their website, the Movies table includes a search/filter page where the user can enter the movie name and director name (and I assume a SELECT query will find the search results?).
* Yes, the movie table has a search/filter page.
* Yes it does

**Does the UI implement an INSERT for every table in the schema? In other words, there should be UI input fields that correspond to each table and attribute in that table.**

* Every page has an insert page they can go to
* Yes - Their website has a separate page for INSERT for each of the five tables. For example, creating an account will use INSERT, adding movies, ratings, directors, and actors seem to utilize INSERT in these pages.
* . Yes, every table has an insert/add page.
* Yes, I see an insert option under each table

**Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship? In other words if there is a M:M relationship between Orders and Products, INSERTing a new Order (e.g. orderID, customerID, date, total), should also INSERT row(s) in the intersection table, e.g. OrderDetails (orderID, productID, qty, price and line\_total).**

* Yes, the pages with M:M have a proper entry to add the FK.
* Yes - Each INSERT adds the corresponding FK attributes. For example, the M:M relationship between Movies and Actors through the actors\_works intersection table, in Movies the FK is actorID and in Actors, the FK is movieID.
* Yes, the actor and movie M:M relationship can be inserted from the actor insert page where users can add movie and director for a specific actor.
* Yes, I see when I INSERT, there is an option insert FK attributes. For example, the INSERT for movies has an actor and director field.

**Is there at least one DELETE and does at least one DELETE remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.**

* Yes, every entity has a delete option. I'm assuming this means hypothetically because this is the front end so there should be no 'usable' delete buttons. But they have all the deletes covered.
* Yes - Another M:M relationship in their website is between Directors and Movies. For this, the Directors table have a DELETE webpage where the user is given a delete button for each entry (i.e. different directors).
* Yes, from the actor insert page, users can either delete movies or directors But the actor won't be deleted. Likewise, deleting either movie or director won't delete the actor, but the M:M relation between actor & movie or actor & director.
* Yes, both actors and directors have a DELETE option. The actor DELETE removes from the M:M relationship of actors to movies

**Is there at least one UPDATE for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?**

* Yes, every entity has an update element.
* Yes - Similar to the above, the M:M relationship between Directors & Movies, besides the delete buttons in the delete page, there are also update (i.e. EDIT) buttons as well. It would appear that delete and update shares the same webpage.
* Yes, ratings, directors, and actors have an update button for each row.
* Yes, I see an UPDATE for ratings, directors, and actors

**Is at least one relationship NULLable? In other words, there should be at least one optional relationship, e.g. having an Employee might be optional for any Order. Thus it should be feasible to edit an Order and change the value of Employee to be empty.**

* Yes, the reviewText does not have to have anything, so a reviewer can simply apply stars with no text.
* Yes - In their Ratings table, the attribute "review" that has the text data type did not indicate a "not NULL" constraint. So this could mean that a Reviewer may be optional for any Ratings.
* Yes, ratings can have null reviewers and null movies
* Yes, the body of a review can be left blank, resulting in a NULL

**Do you have any other suggestions for the team to help with their HTML UI?**

* Yes, I think your landing page style is unnecessary. You can implement everything under each heading onto one page (e.g. you have a movie page where you can read/display, search/filter, and insert etc.) as it stands now it's a little tedious to navigate and I think you will be adding an immense amount of unnecessary work by separating everything like that. I do really like how your individual pages are coming along. They look very slick so far! Great work!
* I think some of the webpages for some tables may be combined into a single page. For example, in Movies, maybe READ/DISPLAY movie, SEARCH/FILTER movie, and INSERT info for a movie could be together? And possibly for Ratings, the READ/DISPLAY rating and INSERT and/or UPDATE rating could be a single page as well?
* I think my suggestion isn't quite related to the course material, but displaying names of actors or directors into first/last names would help users to distinguish foreign actors/dirstors' first/last name.
* For me, the process flow and wording on the homepage is a little hard to read. Splitting into INSERT, DELETE/UPDATE, and SELECT rather than by attribute could make this better, or changing the wording on the individual links. Looks great overall!

**Actions based on the feedback:**

Most of our feedback suggested we combine some pages and to improve the layout of our landing page. We have added a navigation bar to better cycle through our pages and display each entity. Daniel Yu suggested we place the SEARCH/FILTER together with the READ/DISPLAY of “movie”, but we took it a step further and added the search to our navigation bar.