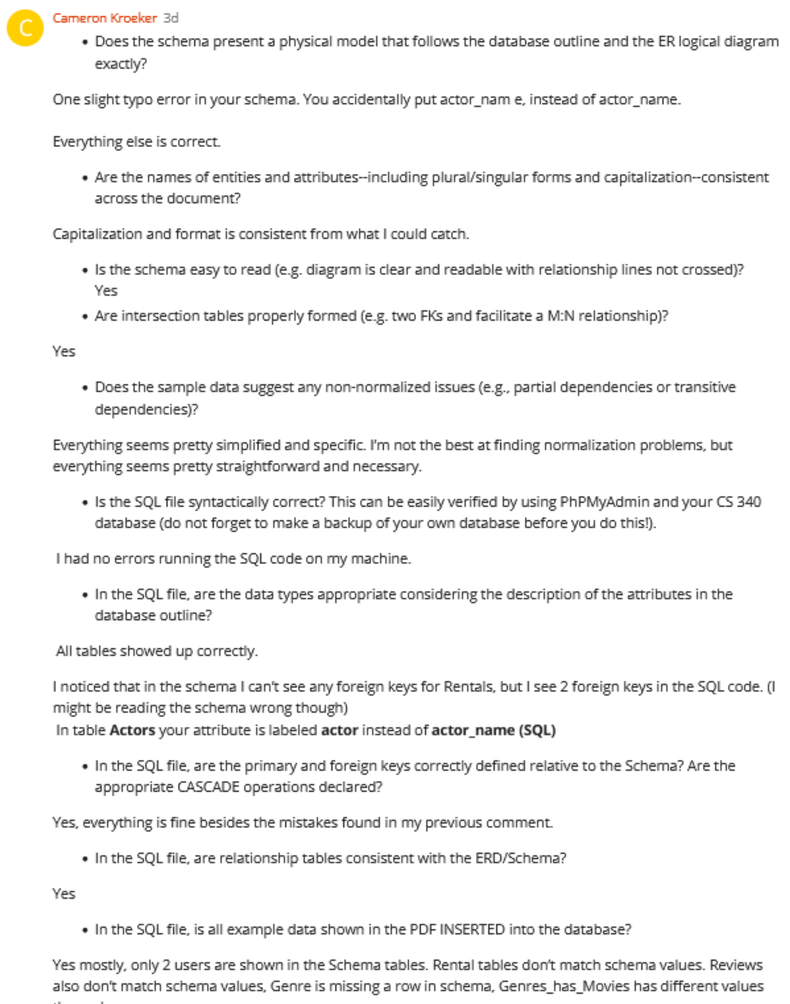
Executive Summary

We have been working to incorporate the feedback we have received into our database. We have been continuing to make more pages for each of the tables all functional and making the attributes in those tables accessible in appropriate ways. One of the biggest changes our database went through was the normalization process. The earlier iterations of our schema had a lot of redundancies but after normalizing it the schema is a lot more streamlined. There were also some changes to attributes when we started implementing the web interface for the database. We realized some attributes weren't necessary and were removed along with some foreign keys being changed to make more sense during implementation.

Over all the content over our database has mostly stayed the same throughout iterations but our schema changed a lot in the early steps because some relations were wrong and/or didn't make sense.

URL: http://flip2.engr.oregonstate.edu:55001/ (make sure to be on osu VPN!)

Feedback



Cameron Kroeker 3d

- Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

One slight typo error in your schema. You accidentally put actor_nam e, instead of actor_name.

Everything else is correct.

- Are the names of entities and attributes--including plural/singular forms and capitalization--consistent across the document?

Capitalization and format is consistent from what I could catch.

- Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)? Yes
- Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?

Yes

- Does the sample data suggest any non-normalized issues (e.g., partial dependencies or transitive dependencies)?

Everything seems pretty simplified and specific. I'm not the best at finding normalization problems, but everything seems pretty straightforward and necessary.

- Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to make a backup of your own database before you do this!).

I had no errors running the SQL code on my machine.

- In the SQL file, are the data types appropriate considering the description of the attributes in the database outline?

All tables showed up correctly.

I noticed that in the schema I can't see any foreign keys for Rentals, but I see 2 foreign keys in the SQL code. (I might be reading the schema wrong though)
In table **Actors** your attribute is labeled **actor** instead of **actor_name (SQL)**

- In the SQL file, are the primary and foreign keys correctly defined relative to the Schema? Are the appropriate CASCADE operations declared?

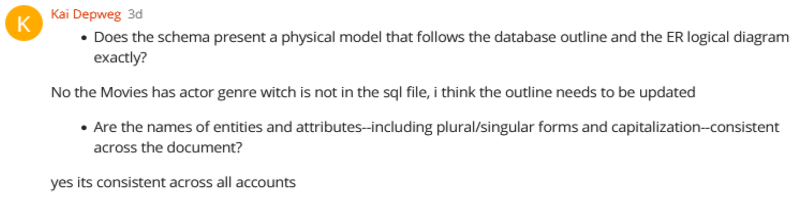
Yes, everything is fine besides the mistakes found in my previous comment.

- In the SQL file, are relationship tables consistent with the ERD/Schema?

Yes

- In the SQL file, is all example data shown in the PDF INSERTED into the database?

Yes mostly, only 2 users are shown in the Schema tables. Rental tables don't match schema values. Reviews also don't match schema values, Genre is missing a row in schema, Genres_has_Movies has different values



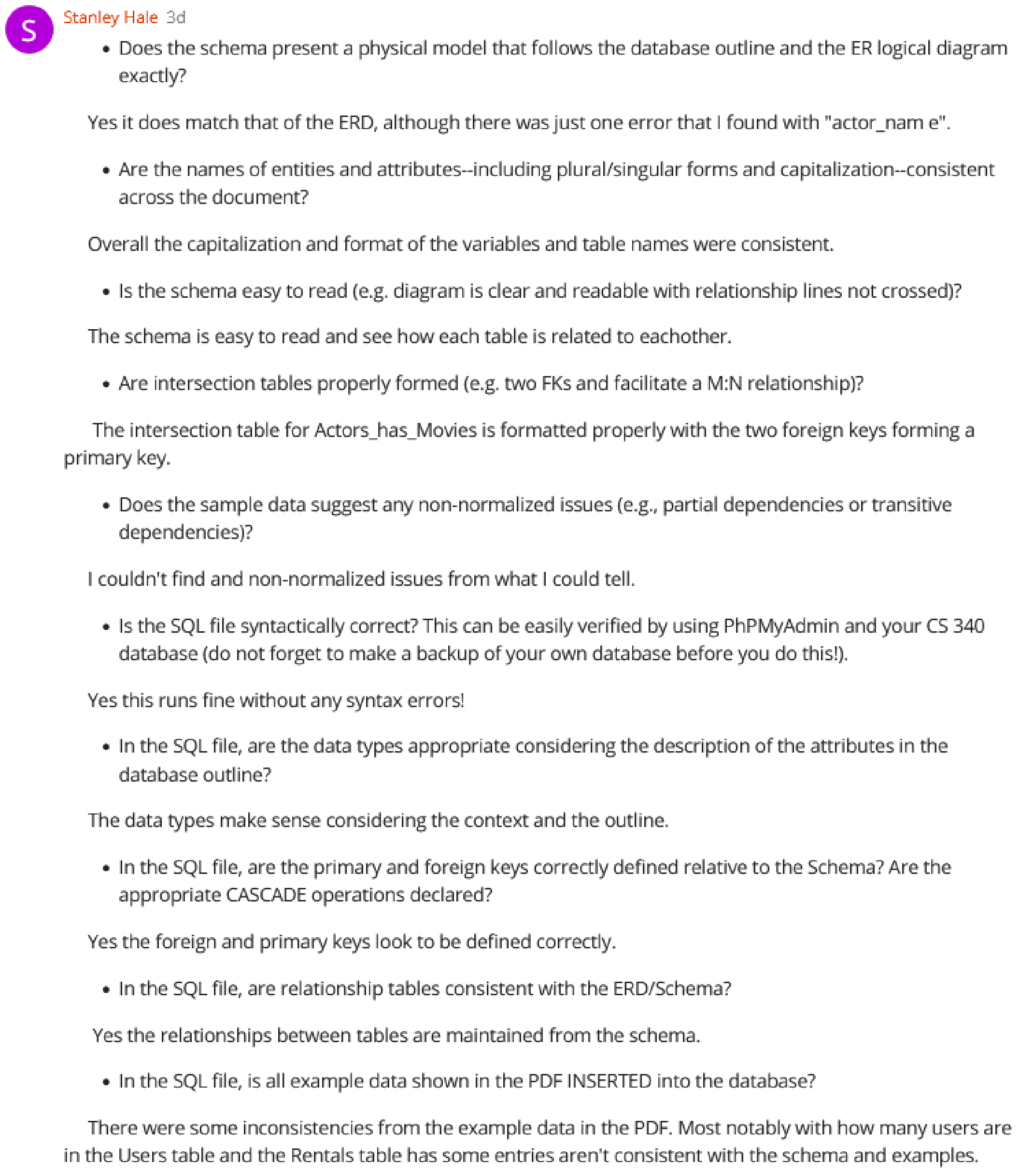
Kai Depweg 3d

- Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

No the Movies has actor genre witch is not in the sql file, i think the outline needs to be updated

- Are the names of entities and attributes--including plural/singular forms and capitalization--consistent across the document?

yes its consistent across all accounts



S Stanley Hale 3d

- Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

Yes it does match that of the ERD, although there was just one error that I found with "actor_nam e".

- Are the names of entities and attributes--including plural/singular forms and capitalization--consistent across the document?

Overall the capitalization and format of the variables and table names were consistent.

- Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?

The schema is easy to read and see how each table is related to eachother.

- Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?

The intersection table for Actors_has_Movies is formatted properly with the two foreign keys forming a primary key.

- Does the sample data suggest any non-normalized issues (e.g., partial dependencies or transitive dependencies)?

I couldn't find and non-normalized issues from what I could tell.

- Is the SQL file syntactically correct? This can be easily verified by using PhPMyAdmin and your CS 340 database (do not forget to make a backup of your own database before you do this!).

Yes this runs fine without any syntax errors!

- In the SQL file, are the data types appropriate considering the description of the attributes in the database outline?

The data types make sense considering the context and the outline.

- In the SQL file, are the primary and foreign keys correctly defined relative to the Schema? Are the appropriate CASCADE operations declared?

Yes the foreign and primary keys look to be defined correctly.

- In the SQL file, are relationship tables consistent with the ERD/Schema?

Yes the relationships between tables are maintained from the schema.

- In the SQL file, is all example data shown in the PDF INSERTED into the database?

There were some inconsistencies from the example data in the PDF. Most notably with how many users are in the Users table and the Rentals table has some entries aren't consistent with the schema and examples.

Step 3 Feedback

P

Paul Lipp 3d

- *Does the UI utilize a SELECT for every table in the schema?* Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).

The UI looks great and is very intuitive and easy to navigate. The HTML elements make sense and the page is overall well done. There is a select for each table on the tables corresponding page.

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

There is no filter implemented at the moment

- *Does the UI implement an INSERT for every table in the schema?* There should be UI input fields that correspond to each table and attribute in that table.

There are inserts for each table

- *Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?* For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

I don't actually see the M:N movies from the pdf implemented (genre_movies, actor_movies), so I can't speak on if the INSERTS work as intended.

- *Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?* For example, 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.

There is a delete for each table, but again the M:N table doesn't seem to be implemented.

- *Is there at least one UPDATE for any one entity?* For example, in the case of Products, can the attributes of an existing row/record be updated?

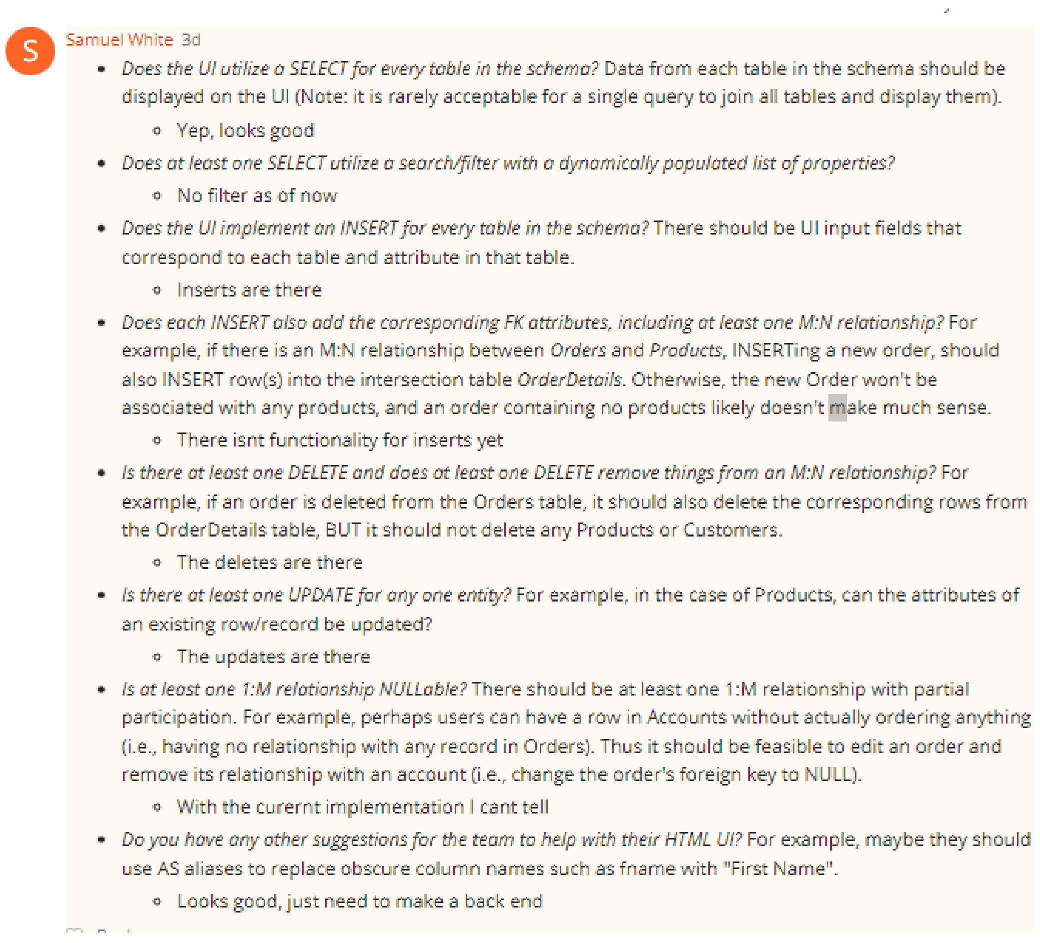
There is an update for each entity.

- *Is at least one 1:M relationship NULLable?* There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).

The UI doesn't indicate whether the 1:M relationships are nullable, and whether the fk's can be set to null, but there are partial participation 1:M relationships, such as genre / movie, or actor / movie, or user / rentals.

- *Do you have any other suggestions for the team to help with their HTML UI?* For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".

Nothing UI related, just implementing the M:N tables if they're still relevant. If not, updating the PDF would help with reviewing future submissions. Excellent choice of films by the way.



Samuel White 3d

- *Does the UI utilize a SELECT for every table in the schema?* Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).
  - Yep, looks good
- *Does at least one SELECT utilize a search/filter with a dynamically populated list of properties?*
  - No filter as of now
- *Does the UI implement an INSERT for every table in the schema?* There should be UI input fields that correspond to each table and attribute in that table.
  - Inserts are there
- *Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?* For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.
  - There isnt functionality for inserts yet
- *Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?* For example, 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.
  - The deletes are there
- *Is there at least one UPDATE for any one entity?* For example, in the case of Products, can the attributes of an existing row/record be updated?
  - The updates are there
- *Is at least one 1:M relationship NULLable?* There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).
  - With the curernt implementation I cant tell
- *Do you have any other suggestions for the team to help with their HTML UI?* For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".
  - Looks good, just need to make a back end

George Hutchinson
3 days ago

- *Does the UI utilize a SELECT for every table in the schema?* Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).

I do see a select for each table as the data is visible.

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

I dont see an ability to search for specific elements with th HTML UI

- *Does the UI implement an INSERT for every table in the schema?* There should be UI input fields that correspond to each table and attribute in that table.

I do see insert ability to add new elements to each table.

- *Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?* For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

According to the schema, there is an M: N relationship but the functionality for the INSERT does not yet exist on the table so I cant tell if it updates it properly.

- *Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?* For example, 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.

Same as the insert I see the ability to do it, but the functionality is not there.

- *Is there at least one UPDATE for any one entity?* For example, in the case of Products, can the attribute: of an existing row/record be updated?

I do see the ability to update an entity.

- *Is at least one 1:M relationship NULLable?* There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).

In theory, it should be possible for a NULLable relationship, and it will be testable when the website has full functionality.

- *Do you have any other suggestions for the team to help with their HTML UI?* For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".

I think the website is quite nice, but in the rentable table when you add a new element it requires a number instead of a name for who is renting it. I would suggest allowing a user to type in their name.

K

Kai Depweg

3 days ago

- *Does the UI utilize a SELECT for every table in the schema?* Data from each table in the schema should be displayed on the UI (Note: it is rarely acceptable for a single query to join all tables and display them).

There is a select corresponding to each page.

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

There is no filter implemented

- *Does the UI implement an INSERT for every table in the schema?* There should be UI input fields that correspond to each table and attribute in that table

.yes we can add new Insert for every table

- *Does each INSERT also add the corresponding FK attributes, including at least one M:N relationship?* For example, if there is an M:N relationship between *Orders* and *Products*, INSERTing a new order, should also INSERT row(s) into the intersection table *OrderDetails*. Otherwise, the new Order won't be associated with any products, and an order containing no products likely doesn't make much sense.

I Actors_movies does not exist in the DML but does have reference in the DDL so no there is no insert into the table

- *Is there at least one DELETE and does at least one DELETE remove things from an M:N relationship?* For example, 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.

again table exists in the DDL not the DML ref

- *Is there at least one UPDATE for any one entity?* For example, in the case of Products, can the attributes of an existing row/record be updated?

Yes there is an update for at least one update in any one entity

- *Is at least one 1:M relationship NULLable?* There should be at least one 1:M relationship with partial participation. For example, perhaps users can have a row in Accounts without actually ordering anything (i.e., having no relationship with any record in Orders). Thus it should be feasible to edit an order and remove its relationship with an account (i.e., change the order's foreign key to NULL).

Yes there exists a table where there is a possible nullable relationship

- *Do you have any other suggestions for the team to help with their HTML UI?* For example, maybe they should use AS aliases to replace obscure column names such as fname with "First Name".

I think simple just add the tables to you DML and then maybe block your code so its simpler to read, much like you did in the DDL where you have commented off sections. I think your website is really good, I like the conformations when deleting, i think also adding the correct conformation message info would be good too.

Actions Based on Feedback: Renamed the composite tables (got rid of the “has” part), crossed out entities that aren’t being used anymore, updated the database outline to better match the schema, and fixed some spelling errors (actor_name is spelled correctly but MySQL just makes it look like there’s a space there).

Actions Based on Feedback (Step 3): We updated the html to include the junction tables for our m:n relationships and we implemented a filter that will filter movies by their genre in the DML, not yet implemented in the html.

Project Title: 2 Block 2 Buster

Team Members: Tanner Parks, Kyle Werstlein

Overview: After almost 15 years Blockbuster has finally realized “Hey maybe the internet isn’t just a fad” and has decided to embrace it. Blockbuster Video was once a dominant player in the movie industry until the rise of streaming services like Netflix and Hulu led to their decline. Fortunately for Blockbuster, streaming services are a dime a dozen today and people are getting sick of having all their favorite shows and movies spread out over half a dozen different services. That’s where Blockbuster comes in. Since they're a movie rental company - not a streaming service; it doesn’t matter which studio owns the rights to what so you can find all your favorite movies in one place. Blockbuster is looking to create a new platform where their users will have access to a large library of titles that they can browse and rent online, as well as features like personalized recommendations, and user reviews.

Database Outline:

Movies: records information about movies

- movieID: int, auto_increment, unique, not NULL, Primary Key
- title: varchar(45), not NULL
- ~~actor: varchar(45), not NULL~~
- ~~genre: varchar(45)char, not NULL~~
- description: longtext
- duration: int, not NULL

Users: records information about the users who submit reviews

- userID : int, auto_increment, unique, not NULL, Primary Key
- password: varchar(45), not NULL
- first_name: varchar(45), not NULL
- ~~last_name: varchar(45), not NULL~~
- email: varchar(45), not NULL

Rentals: records information about rental histories

- rentalID: int, auto_increment, unique, not NULL, Primary Key
- userID : int, not NULL, Foreign Key to Users
- movieID: int, not NULL, Foreign Key to Movies
- rental_date: date, not NULL
- ~~return_date: date, not NULL~~

Reviews: records information about user movie reviews

- reviewID: int, auto_increment, unique, not NULL, Primary Key
- userID : int, not NULL, Foreign Key to Users

- movieID: int, not NULL, Foreign Key to Movies
- rating: float
- ~~num_ratings: int, not NULL~~

Actors: Top billed actors in movies

- actorID: int, auto_increment, unique, not NULL, Primary Key
- actor_name: varchar(45), not NULL

Actors_Movies: Movies of actor

- actorID: int, auto_increment, unique, not NULL, Foreign Key to Actors
- movieID: int, auto_increment, unique, not NULL, Foreign Key to Movies

Genres: records information about movie genres

- genreID: int, auto_increment, unique, not NULL, Primary Key
- genre_name: varchar(45), not NULL

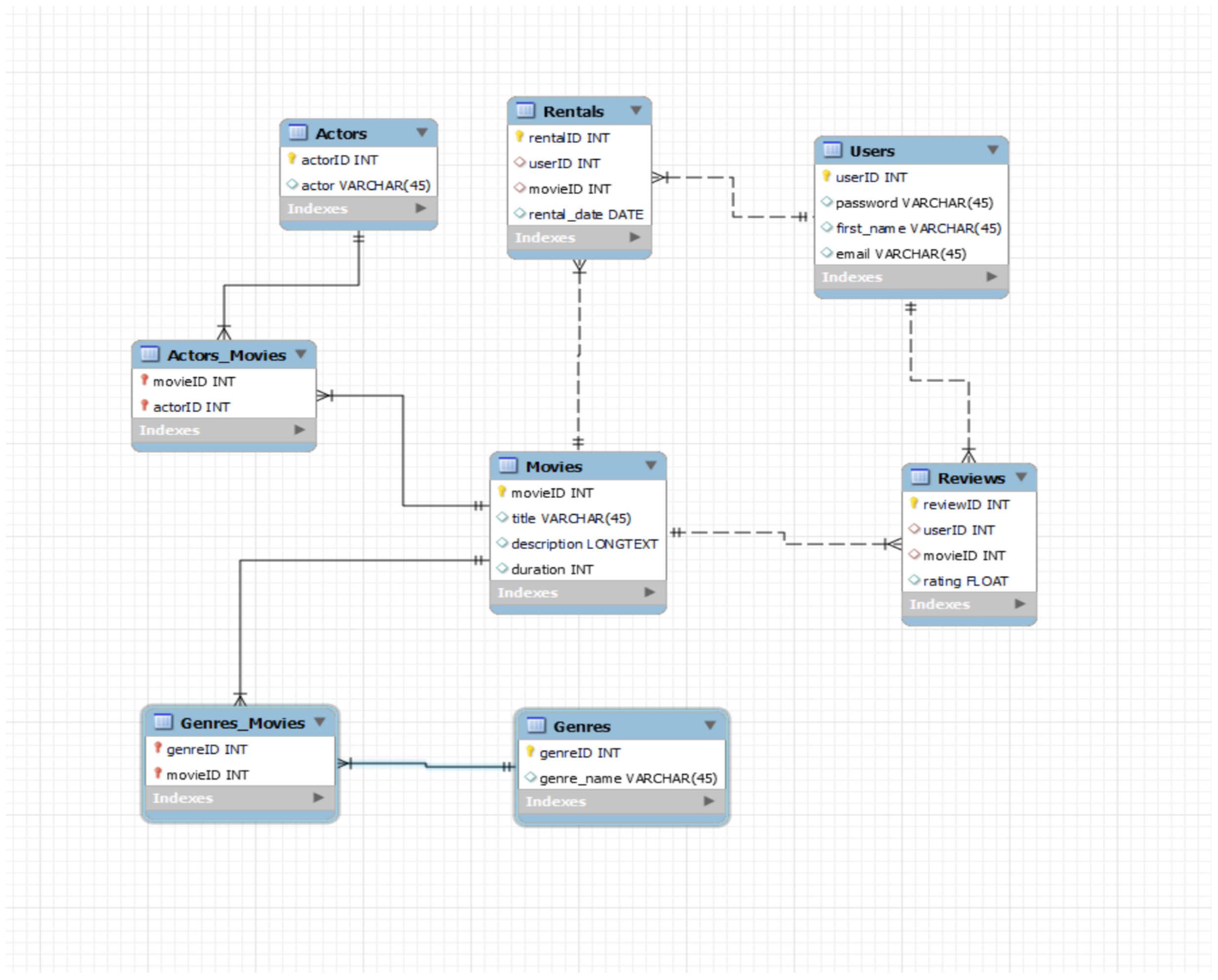
Genres_Movies: a junction table that links movies and genres

- movieID: int, not NULL, Foreign Key to Movies
- genreID: int, not NULL, Foreign Key to Genres

Relationships:

- a M:1 relationship between **Movies and Rentals** is implemented with movieID as a foreign key inside of Rentals
- a 1:M relationship between **Movies and Reviews** is implemented with movieID as a foreign key inside of Reviews
- a 1:M relationship between **Users and Rentals** is implemented with userID as a foreign key inside of Rentals

- a 1:M relationship between **Users and Reviews** is implemented with userID as a foreign key inside of Reviews
- A M:N relationship between **Movies and Genres** is implemented with a junction table connecting the two tables so a movie can have multiple genres, and a genre can be associated with multiple movies
- A M:N relationship between **Movies and Actors** is implemented with a junction table connecting the two tables so a movie can have multiple actors, and an actor can be associated with multiple movies



Schema updated to move actors into their own table after normalization.

Example data:

Movies table:

movieID	**title**	**description**	**duration**
1	The Shawshank Redemption	Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.	142
2	The Godfather	The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son.	175
3	The Dark Knight	When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.	152

Genres table:

genreID	genre_name
1	Drama
2	Crime
3	Action
4	Romance

Genres_Movies table

genreID	movieID
1	1

2	2
3	3

Rentals table

rentalID	userID	movieID	rental_date
1	1	1	2022-03-15
2	1	3	2022-03-15
3	2	1	2022-03-16

Users Table

userID	password	first_name	email
1	password1	John	john@example.com
2	password2	Jane	jane@example.com
3	BillyRockz1!	Billy	BigBill@gmail.com

Reviews Table

reviewID	userID	movieID	rating	~~num_ratings~~
1	1	1	9.3	~~1000~~
2	1	2	9.0	~~800~~
3	2	3	8.8	~~700~~

Actors

actorID	actor_name
1	Tim Robbins
2	Morgan Freeman
3	Marlon Brando
4	Al Pacino
5	Christian Bale
6	Heath Ledger

Actors_Movies

movieID	actor
1	Tim Robbins
1	Morgan Freeman
2	Marlon Brando
2	Al Pacino
3	Christian Bale
3	Heath Ledger

Step 4:

Currently only the movies table is connected to the database and functioning and the buttons work and the changes are reflected in the phpMyAdmin panel.