COS 221 - Practical Assignment 4

Armand Krynauw u04868286

May 3, 2022

Contents

Task 1: Obtain the Sakila database

See Task 2 for more information.

Task 2: E(E)R-diagram

2.1 - E(E)R Diagram

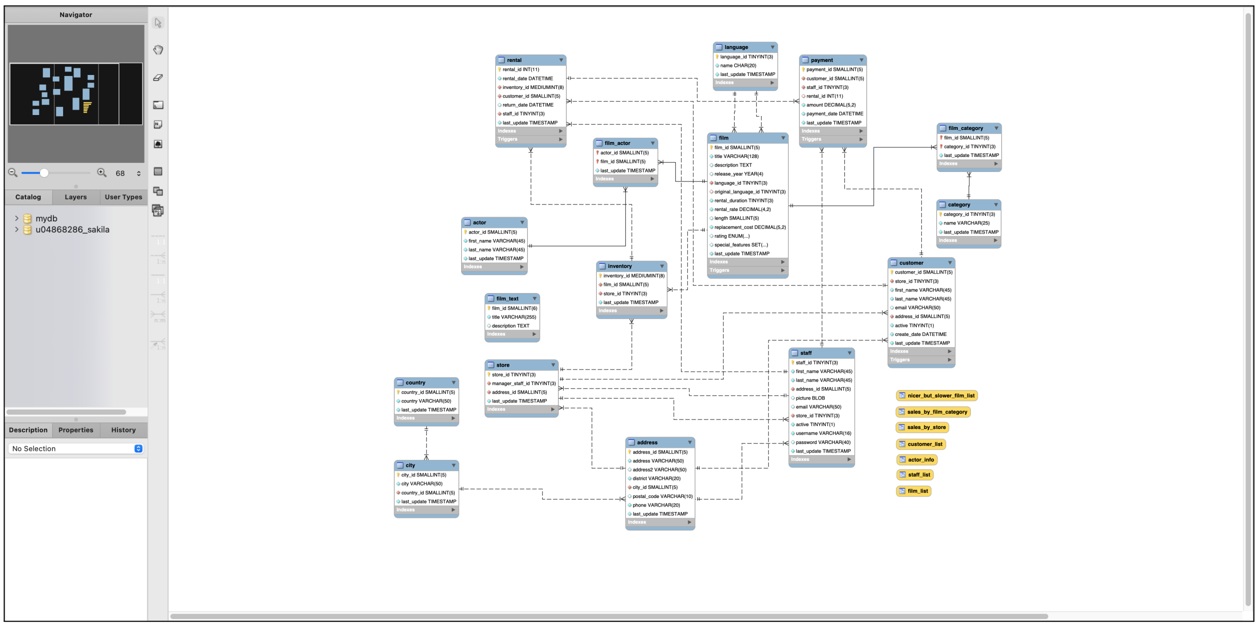

Figure 1: E(E)R-diagram of the Sakila database generated by MySQL Workbench

2.2 - Exploring store table structure

The **store** table stores the *store_id*, *manager_staff_id*, *address_id* and *last_update*.

The *store_id* attribute is the primary key which stores the ID of the store and has a data type of TINYINT which can store 255 as its maximum unsigned value. The *manager_staff_id* attribute has a data type of TINYINT as well. The *manager_staff_id* attribute is a foreign key which relates the manager of each store to the **staff** table by storing the staff ID of the store's manager. The *address_id* attribute has a datatype of SMALLINT which can store 65535 as its maximum unsigned value. The *address_id* attribute is a foreign key which relates each store to its address in the **address** table. The *last_update* attribute has a data type of TIMESTAMP which stores Unix Epoch time as a value with date and time parts. When a new record is created *last_update* will be set to the current timestamp and every time the record gets updated it will be set to the current timestamp.

None of the attributes can be NULL. The *manager_staff_id* attribute id unique and the *store_id* attribute is auto incremented.

Task 3: Maven Managed Project

See the **pom.xml** file in the project for the addition of the required properties and tags.

Also, see the README file which outlines the steps to get the project running.

Check out the following link for a short demo of the final project: **Practical 4 Demo**

Task 4: Graphical User Interface

4.1 - Tabbed Panes

See the application interface for the creation of the necessary tabbed panes.

4.2 - Displaying staff data

Nothing special was done for this task. The staff data is just displayed as required.

4.3 - Filtering staff results

The specification was ambiguous about the type of filter which has to be implemented. For this reason I chose to implement a general purpose filter where a user can select by which column they want to filter their searchers. There is also an option to filter the entire database.

4.4 - Adding a new film to the database

I decided to include input validation for new film data. The input validation is implemented on the client side. The validation is 'hard coded' meaning that the validation occurs before trying to insert anything in the database. This was done with the intent to cut down on complex logic but to still have some sort of input validation. It was also assumed that the film schema is unlikely to change in the future.

It was decided to not display the description field in the table since the content does not fit the table cells.

4.5 - Store inventory

Nothing special was done for this task. The store inventory is just displayed as required.

4.6 - Adding, updating, and deleting customers from the database

I also decided to include input validation for customer operations. The validation implemented was a mixture of client side and server side validation. On the client side the data was mainly validated for the correct data length and type. On the server side validation was done to ensure that the city and store ID is valid. It was assumed that the city of the customer already has to be registered in the database for it to be valid.

When deleting a customer their rental information will be kept in some of the other database tables if they had any. This is due to the fact that there are no triggers set up in the database schema to cater for this. It was thought best to leave this as it is. If the database administrator would like to have the deleted customer's data removed from all tables then it will be best to implement triggers in the schema and not the 'client side' code.

4.7 - Extra

All the code was manually written. No UI designer was used.

Task 5: Bonus Marks

5.1 - Environment variables

I did make use of environment variables in this project to connect to my database. The method of importing the environment variables can be found in the top of my **Database** class in my project.

5.2 - Git and GitHub

I made use of Git to do regular pushes of my codebase to GitHub while developing this project. The online repository can be found at the following link: **Practical 4 Repository**

Note, at the time of writing this document the repository is private for security reasons. It will be made public after the submission deadline.

5.3 - Advanced SQL

No use was made of any advanced SQL.