Development of A Web Application for MUBI Movie Lovers

--Part 2: MUBI Database Design

Team

Jie Zhen, jz67@iu.edu

Xuemei Hu, xh18@iu.edu

Xin Tu, xintu@iu.edu

Summary

Relational databases store data in rows and columns in tables (or relations). The relationships that can be created among the tables enable a relational database to efficiently store huge amounts of data, and effectively retrieve selected data. In this part, we used Structured Query Language (SQL) to generate the MUBI database model with optimized tables and relationships. Firstly, we created the MUBI database and imported data into it using queries in MySQL workbench. Secondly, we defined the database entities and reconstructed the tables by choosing attributes and defining data types and constraints. Thirdly, we defined the primary keys and foreign keys and created the relationships among tables. Entity Relationship (ER) diagram showed the relationship of entity sets stored in the MUBI database. Our database design will enable a relational MUBI database to efficiently extract data from the database and provide the back-end database for Shiny app connection.

Database Design

Data import

We used MySQL Workbench to perform the MUBI database design. The local SQL server was created by using MAMP and was connected to MySQL Workbench. We created a new database called MUBI database. We downloaded the original MUBI CSV files and imported the table data from the CSV files using import wizard (Figure 1). Large MUBI CSV files were imported by load data infile queries (Figure 1).

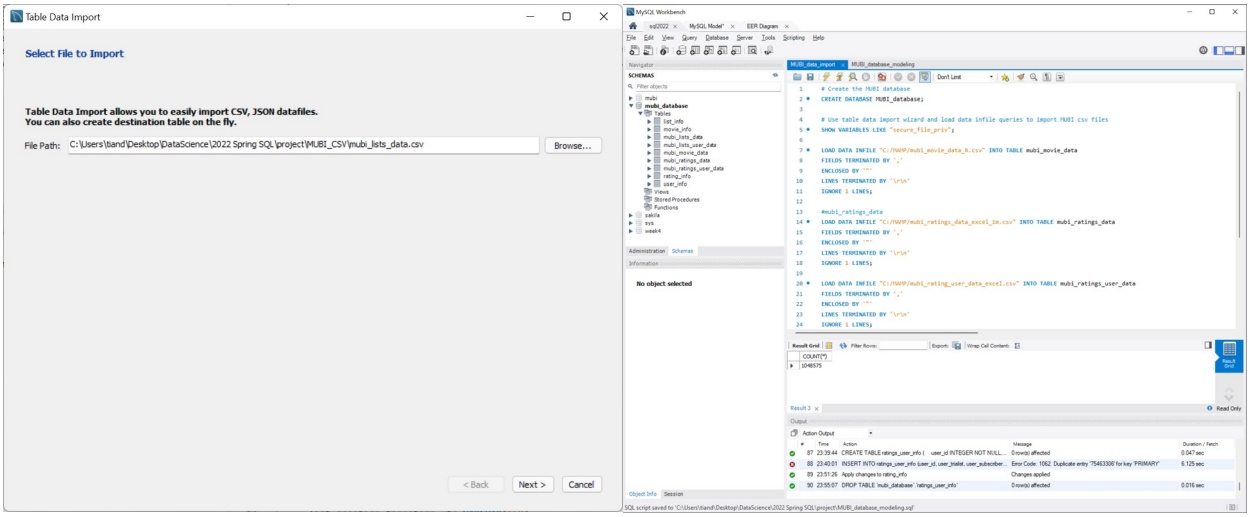

Figure 1. Import MUBI CSV files using (**left**) table data import wizard and (**right**) MySQL queries

MUBI database modeling

We examined the MUBI data tables and found that there were four entities: lists, users, ratings, and movies. We restructured the MUBI tables into subject-based tables for each entity (Figure 2). There were two steps to generated tables. Firstly, we used CREATE queries to create blank tables with specified columns. Secondly, we used INSERT queries to insert selected values into tables. Codes examples were showed in Figure 2. Four tables were created using this method. The list_info table contained the lists information and foreign keys. The rating_info table is the fact table containing rating fact and foreign keys. The user_info table and movie_info table are dimension tables storing the descriptive attributes, which can be used to filter, categorize, and label data. For the lists, we stored the ID of the MUBI list, the ID of the user, title, the number of movies

in the list, the update and creation time, the number of followers, the URL of the list, the comment of the list, and the image URLs of the list. In the rating table, we stored the rating ID, URL, score, time, movie ID, and user ID. In the user table, we stored information including the user ID, trial or not, subscriber or not, eligible for trial or not, and having payment method or not. For the movie table, it contained movie ID, title, the year of releasing, URL, the language of title, and popularity scores.

Figure 2. Database modeling: (**left**) queries for creating tables, (**right**) four created tables

Define data types and constraints

Each column in the MUBI tables has a column name and a specified data type. The data type is a guideline for the MUBI database to understand what type of data is expected inside of each column, and it also identifies how MBUI will interact with the stored data. There are three main data types specified in the MUBI data: string, numeric, and date and time. In the MUBI SQL database, we used these data types: integer, variable character, text, and date. All the primary keys (user_id, rating_id, list_id, movie_id) in four tables were defined as integers. All the columns contain letters, numbers, and special characters used data type variable characters. These columns were user_trialist, user_subscribe, user_eligible_for_trial and user_has_paymet_method in table

user_info; In table rating_info, column rating_usl and rating_timestamp_utc; In table movie_info, column moive_title, movie_usle and movie_title_lanague.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted. Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table. There are some commonly used constraints in MySQL including Not Null, uniques, primary key, foreign key, check, auto increment, date, check, date and default. Here, we created four tables including user_info, list_info, movie_info and movie_info, and defined the following constraints. The user_info table uses Not Null for the whole table, Auto_Increment and Not Null for the primary key user_id. The rating_info table uses Not Null for the whole table, primary key for rating_id. The list_info table uses Not all for the whole table, primary key for the list_id, foreign key and Unique for the user_id. The movie_info table uses Not Null for the whole table, primary key for the movie_id.

Define primary and foreign keys

Each table has its own primary key variable, which is a unique identifier of each record in the table. Each table also has foreign keys to link to other tables. In the list_info table, we defined list_id as the primary key and user_id as the foreign key. In the rating_info table, rating_id was the primary key and movie_id and user_id were foreign keys. In the user_info table, user_id was the primary key. In the movie_table table, we used movie_id as the primary key.

Entity Relationship (ER) Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates the relationship of entity sets stored in a database. We used two tools to generate an ER diagram for the MUBI database model. One was created by draw.io (Figure 3 left panel) and the other was generated by MySQL workbench (Figure 3 right panel). As we described before, our MUBI database has 4 entities: user information (user_info), list information (list_info), rating information (rating_info), and movie

information (movie_info). According to DBMS theories, each table has its own primary key variable, which is a unique identifier of each record in the table. Each table also has foreign keys to link to other tables. The ER diagram of the MUBI database showed that the attributes (columns) of each table, the primary key and foreign keys of each table, and the relationships among the tables. The dashed lines indicate that two tables can be linked through a foreign key. In the list_info table, we defined list_id as the primary key and user_id as the foreign key. So, it can be linked to the user_info table through the foreign key user_id. In the rating_info table, rating_id was the primary key and movie_id and user_id were foreign keys. It can be linked to the user_info table through the foreign key user_id and linked to the movie_info table through the foreign key movie_id. All of these relationships were one or many-to-one relationships. These relationships included in the ER diagram suggested that each list was generated by different users and users also had different rating facts for different movies.

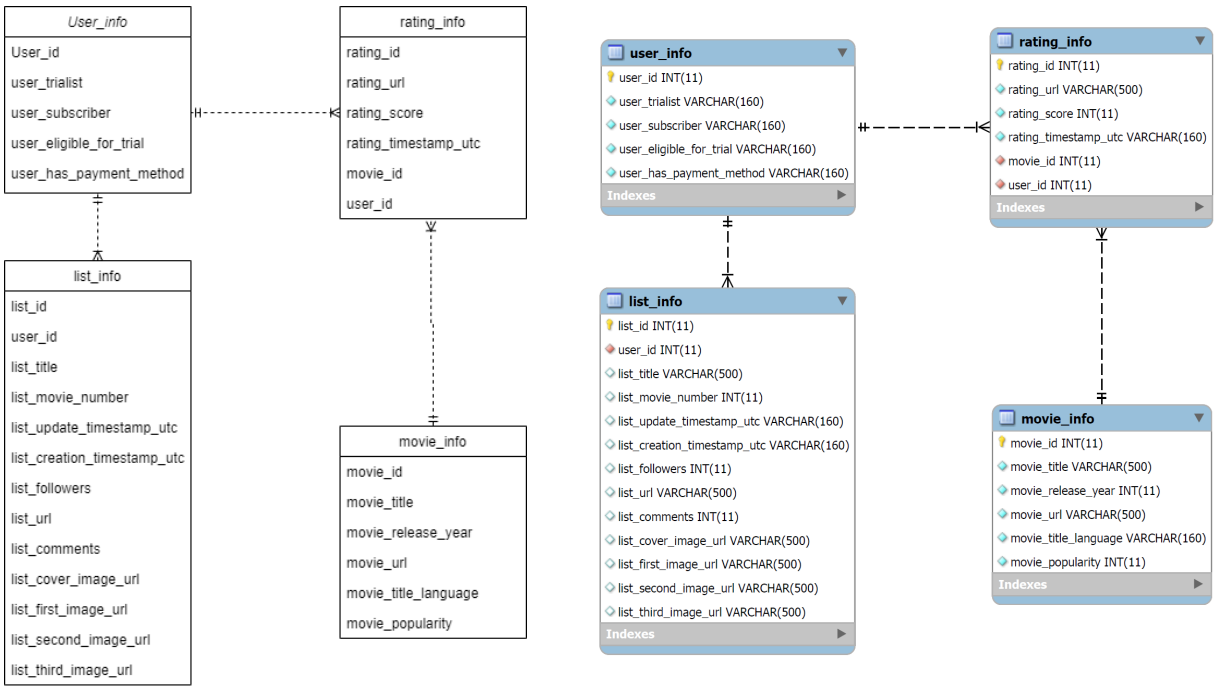

Figure 3. The Entity Relationship (ER) diagram created by (**left**) draw.io and (**right**) MySQL Workbench

Data dictionary with metadata

user_info

Column Name	Datatype	Primary Key	Not Null	Unique	Foreign Key
user_id	Int(11)	YES	YES	YES	
user_trialist	Varchar(160)		YES		
user_subcriber	Varchar(160)		YES		
user_eligible_for_trial	Varchar(160)		YES		
user_has_payment_method	Varchar(160)		YES		

list_info

Column Name	Datatype	Primary Key	Not Null	Unique	Foreign Key
list_id	Int(11)	YES	YES	YES	
user_id	Int(11)		YES		YES
list_title	Varchar(500)		YES		
list_movie_number	Int(11)		YES		
list_update_timestamp_utc	Varchar(160)		YES		
list_creation_timestamp_utc	Varchar(160)		YES		
list_followers	Int(11)		YES		
list_url	Varchar(500)		YES		
list_comments	Int(11)		YES		
list_cover_image_url	Varchar(500)		YES		
list_first_image_url	Varchar(500)		YES		
list_second_image_url	Varchar(500)		YES		
list_third_image_url	Varchar(500)		YES		

rating_info

Column Name	Datatype	Primary Key	Not Null	Unique	Foreign Key
rating_id	Int(11)	YES	YES	YES	
rating_url	Varchar(500)		YES		
rating_score	Int(11)		YES		
rating_timestamp	TIMESTAMP		YES		
movie_id	Int(11)		YES		YES
user_id	Int(11)		YES		YES

movie_info

Column Name	Datatype	Primary Key	Not Null	Unique	Foreign Key
movie_id	Int(11)	YES	YES	YES	
movie_title	Varchar(500)		YES		
movie_release_year	Int(11)		YES		
movie_url	Varchar(500)		YES		
movie_title_language	Varchar(160)		YES		
movie_popularity	Int(11)		YES		

Contributions

Team member	Task contributions
Jie Zhen	Define data types and constraints
Xuemei Hu	Define primary keys and foreign keys, create data dictionary with metadata
Xin Tu	Data import, database modeling, create ER diagram, and report

Future work

Next, we will use Shiny to build an interactive web application with basic functions by connecting to the MUBI database. We will also work on the back end to improve our web application with full user functionalities. Our project will provide the MUBI movie lovers a web tool to access, manage, and analyze the MUBI movie data.

References

https://shanghai.hosting.nyu.edu/data/r/case-3-sql-shiny.html

https://forums.mysql.com/read.php?152,674208,674208

https://www3.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html

https://www.lucidchart.com/pages/er-diagrams