Movie Database Analysis

Heewon Yang, Junghoon Choi, Heeseung moon, Wonbin Shin

*Computational and Data Sciences, George Mason University*

*hyang40@gmu.edu, jchoi202@gmu.edu, hmoon22@gmu.edu, wshin3@gmu.edu*

Keywords: Movie, Hollywood, IMDB, Rating, Actor, Director, Genre, Movie\_actor, Movie\_Genre, Review

Abstract: This paper explores the design and implementation of a Movie Database Management System (DBMS) for managing and querying large-scale movie-related data. The system is built to treat key attributes such as movie titles, genres, release years, ratings, cast, and Director. We discuss the use of SQL queries for efficient data searching. The system's structure is optimized for searching through movie-related data, supporting both small and large datasets. Performance evaluation demonstrates its ability to quickly process complex queries, such as filtering movies by complex queries using key attributes from tables (e.g., genre, rating, release year). In addition, this paper highlights the importance of data integrity and security including reviews and ratings. The proposed Movie DBMS offers preference identification for both academic and commercial applications. serving as a foundation for movie recommendation systems, analysis tools, and digital media libraries. Future work will focus on enhancing the system’s more precise identification of key attributes in movies.

# 1 Introduction

The movie industry developed over decades. Because lots of movies came out, people who don’t know the movie well might have difficulties choosing movies. In consideration of these difficulties, we make this database. Movies consist of various components. When you think of a movie, most people will think of either the main actors or the genre. But movies are made with more people's efforts. Our database covers not only the genre and actors, but also more detailed information about ratings, directors, and actors. The use of this database is to show films that reflect the preferences of the picker. For example, if you choose the actor and director you want, they will show related films.

2 Database schema and design choices

All Tables as csv files are from Kaggle. Kaggle is the world's largest data science community with powerful tools and resources.

2.1 Actor Data

The dataset includes several columns. Our DBMS uses five columns in the Actor table. .Five columns are Actor\_ID, Name, Birth\_Year, Death\_Year, Primary\_profession., The dataset includes information about actors organized into five columns. The primaryName column displays the names of the actors, while the birthYear and deathYear columns indicate their respective years of birth and death. The primary profession column details their main professional roles, encompassing acting, directing, and other contributions to the entertainment industry.Titles are known as the name of a movie. IMDb Actors and Movies” in Kaggle.

2.2 Ratings Data

This section must be in four columns. Four columns are Genre, Rank,RatingTomatometer, Title, No. of Review. Ratings provide rotten tomato ratings and number of reviews.Number of reviews make sure credibility. Titles are known as the name of a movie. The dataset is “Top 100 Rotten Tomatoes movies by genres” in kaggle.

Table shows the ratings about the rotten tomatoes csv file.

2.3 Director Data

The dataset comprises information on 6,820 movies released between 1986 and 2016, averaging approximately 220 movies per year. Each movie is characterized by various attributes, including budget (production costs, with some values represented as 0 for missing data), production company, country of origin, director, main genre, total gross revenue, title, MPAA rating (e.g., R, PG), release date (formatted as YYYY-MM-DD), runtime (in minutes), IMDb user rating, number of user votes, main actor or actress, and writer. This data, scraped from IMDb, serves as a valuable resource for analyzing trends in the movie industry and understanding the factors that contribute to a movie's success.

2.4 Movie Data

This dataset encompasses a range of attributes on films and their critiques. The Rating column reflects the scores assigned by critics, serving as an evaluation of the films. In the Reviews column, you'll find written analyses that offer insights and observations about the films' content and quality. The movie\_name column identifies the specific works being reviewed. The Genres column categorizes the films into different groups, while the Descriptions column presents detailed written analyses. These descriptions convey distinct emotional tones, which are captured in the Emotions column.

2.5 Box Office Collections Data

The dataset contains detailed information about movies, encompassing various attributes related to their performance, cast, and critical reception. It includes columns such as Movie title, Year of release, Director, and Cast, offering essential insights into each film. Moreover, it provides indicators of critical and audience reception through Score, Adjusted Score, IMDB Rating, and metascore, along with the number of Votes received on IMDb. Financial performance is depicted by the Box Office Collection, while the time\_minute column signifies the movie's runtime in minutes. The dataset further classifies each movie by its Imdb\_genre and presents a summary of reviews in the Consensus column. With this array of attributes, the dataset serves as a valuable tool for analyzing the critical and commercial.

**3 SQL code examples and explanations**

**3.1 First Analysis**

We tried to find the titles of movies with an IMDb rating of 8 or higher, or a Metascore of 80 or higher.

SELECT Title

FROM Movie

WHERE IMDB\_Rating >= 8

UNION

SELECT Title

FROM Movie

WHERE Metascore >= 80;

Explanation: The first SELECT statement filters movies based on IMDb ratings, while the second SELECT statement filters movies based on Metascore. The UNION operator combines the results of both queries into a single result set, automatically removing duplicate titles to ensure that each movie appears only once, even if it satisfies both conditions.

**3.2 Second Analysis**

Next, we tried to calculate the number of movies and average IMDb rating for each genre.

SELECT G.Main\_Genre, COUNT(MG.Movie\_ID) AS Movie\_Count, AVG(M.IMDB\_Rating) AS Average\_Rating

FROM Genre G

JOIN Movie\_Genre MG ON G.Genre\_ID = MG.Genre\_ID

JOIN Movie M ON MG.Movie\_ID = M.Movie\_ID

GROUP BY G.Main\_Genre;

Explanation: COUNT counts the number of movies for each genre.

AVG calculates the average IMDb rating for each genre.

Results are grouped by Main\_Genre.

ORDER BY Average\_Rating DESC sorts genres by the highest average IMDb rating first.

**3.3 Third Analysis**

Third, we tried to list genres by total box office earnings, sorted in descending order.

SELECT G.Main\_Genre, SUM(M.Box\_Office) AS Total\_Box\_Office

FROM Genre G

JOIN Movie\_Genre MG ON G.Genre\_ID = MG.Genre\_ID

JOIN Movie M ON MG.Movie\_ID = M.Movie\_ID

GROUP BY G.Main\_Genre

ORDER BY Total\_Box\_Office DESC;

Explanations: This SQL query calculates the total box office revenue for each genre by summing up the Box\_Office values for movies associated with each genre. It uses JOIN operations to link the Genre, Movie\_Genre, and Movie tables. The results are grouped by the Main\_Genre, and the SUM function is applied to calculate the total revenue for each genre. Finally, the results are ordered in descending order of total revenue to display the most profitable genres first.

**3.4 Fourth Analysis**

Fourth, we tried to answer the question: who are the directors with an average movie IMDb rating above 7.5? The code is this

SELECT D.Name AS Director\_Name, AVG(M.IMDB\_Rating) AS Average\_Rating

FROM Director D

JOIN Movie M ON D.Director\_ID = M.Director\_ID

GROUP BY D.Name

HAVING AVG(M.IMDB\_Rating) > 7.5

ORDER BY Average\_Rating DESC;

Explanation: This SQL query calculates the average IMDb rating for each director by joining the Director and Movie tables using their respective IDs. It groups the data by director names and uses the AVG function to compute the average rating for each director. The HAVING clause filters the results to include only directors with an average rating greater than 7.5. Finally, the results are ordered in descending order of average rating to display the highest-rated directors first.

**3.5 Fifth Analysis**

Finally, we tried to find the highest-rated movie in each genre.

WITH GenreMaxRatings AS (

SELECT G.Genre\_ID, MAX(M.IMDB\_Rating) AS Max\_Rating

FROM Genre G

JOIN Movie\_Genre MG ON G.Genre\_ID = MG.Genre\_ID

JOIN Movie M ON MG.Movie\_ID = M.Movie\_ID

GROUP BY G.Genre\_ID

)

SELECT DISTINCT G.Main\_Genre, M.Title, GM.Max\_Rating

FROM Genre G

JOIN Movie\_Genre MG ON G.Genre\_ID = MG.Genre\_ID

JOIN Movie M ON MG.Movie\_ID = M.Movie\_ID

JOIN GenreMaxRatings GM ON G.Genre\_ID = GM.Genre\_ID

WHERE M.IMDB\_Rating = GM.Max\_Rating

ORDER BY GM.Max\_Rating DESC;

This SQL query uses a Common Table Expression (CTE) named GenreMaxRatings to calculate the maximum IMDb rating for each genre by grouping the data by genre ID. In the main query, it joins the Genre, Movie\_Genre, and Movie tables with the GenreMaxRatings CTE to find the movies with the highest rating for each genre. The WHERE clause ensures only the top-rated movies are included, and the results are sorted in descending order of the maximum rating.

**4. ER Modeling and Schema Reduction**

The Entity-Relationship (E-R) model is a conceptual framework used to design and structure movie databases. It provides a visual representation of entities, their attributes, and the relationships between them, allowing for a clear understanding of the database’s architecture. In this model, entities such as Movie, Director, Genre, Actor, and Review form the core components, while relationships like Movie\_Genre and Movie\_Actor define the connections among these entities. This section explores these relationships in detail, emphasizing their cardinalities and implementation in the database schema.

**4.1 ER Modelling**

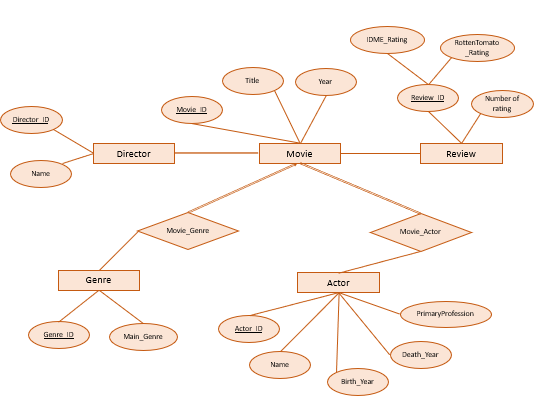
****

Figure Movie Database ER Model

**Figure 1: This caption has one line so it is centered.**

**4.2 Cardinality in the E-R Model Detailed Analysis**

In this section, the cardinalities in the Entity-Relationship (E-R) model are discussed, focusing on key relationships between entities. Each relationship is analyzed in terms of its cardinality, constraints, and implementation in the database schema.

**4.2.1 Director and Movie Relationship**

* Cardinality**:** One-to-Many (1:N)

A single director can direct multiple movies, whereas each movie is associated with only one director. This relationship is implemented by incorporating a foreign key, Director\_ID, in the Movie table, which references the primary key in the Director table. The arrow pointing from Movie to Director in the E-R diagram represents this dependency.

**4.2.2 Actor and Movie Relationship**

* Cardinality: Many-to-Many (M:N)

A single movie can feature multiple actors, and each actor can appear in multiple movies. To manage this many-to-many relationship, an intermediary table, Movie\_Actor is introduced. This linking table contains two foreign keys: Movie\_ID referencing the Movie table and Actor\_ID referencing the Actor table. These foreign keys collectively form the composite primary key for the Movie\_Actor table, ensuring a bidirectional relationship.

**4.2.3 Genre and Movie Relationship**

* Cardinality**:** Many-to-Many (M:N)

A movie can belong to multiple genres, and each genre can include multiple movies. This relationship is managed through a linking table, Movie\_Genre, which contains foreign keys Movie\_ID and Genre\_ID. These keys establish the connection between the Movie and Genre tables, allowing flexible classification of movies into various genres.

**4.2.4 Movie and ReviewRelationship**

* Cardinality: One-to-Many (1:N)

A single movie can have multiple reviews, while each review is linked to only one movie. This relationship is implemented by including a foreign key, Movie\_ID, in the Review table, which references the primary key in the Movie table. The arrow from Review to Movie in the E-R diagram reflects this dependency.

**4.3 Transition to Relational Schema**

Based on this E-R model, the relational schema was created by reducing the relationships and mapping them into actual database tables. This schema is designed to efficiently store and manage data about movies, directors, actors, genres, and reviews.

**4.4 Relational Schema Explanation**

* **Movie Table**: The Movie table, identified by Movie\_ID as the primary key, stores details about each movie and connects to the Director table through a foreign key.
* **Director Table:** The Director table stores information about directors, with Director\_ID as its primary key.
* **Actor Table:** The Actor table contains actor details, and its many-to-many relationship with movies is managed through the Movie\_Actor linking table.
* **Genre Table:** Genres are stored in the Genre table, and the Movie\_Genre table connects movies to their genres.
* **Review Table:** Finally, reviews are managed in the Review table, which links back to the Movie table using the Movie\_ID foreign key.
* **Movie\_Actor Table**: An intermediary table containing foreign keys Movie\_ID and Actor\_ID to manage the many-to-many relationship between movies and actors.
* **Movie\_Genre Table**: Contains foreign keys Movie\_ID and Genre\_ID to represent the many-to-many relationship between movies and genres.

**5 Discussion of Relational Storage and ACID Compliance**

To ensure full ACID compliance in our database, we recommend using a relational database management system (RDBMS) that enforces ACID properties, such as PostgreSQL, MySQL with InnoDB, or SQLite in full transactional mode. Transactions should be defined appropriately for operations involving multiple tables, such as inserting a movie and its related Movie\_Actor and Movie\_Genre rows in a single transaction. The correct isolation level should be selected based on the application's concurrency requirements to balance consistency and performance. Additionally, enabling database backups and write-ahead logging (WAL) ensures durability, safeguarding data against unexpected failures while maintaining integrity.

**6 Challenges faced and solutions**

We had a total of two challenges while we were doing our project. The first one is choosing datasets and combining them. However, we applied techniques such as removing duplicates, standardizing formats, handling missing values, and validating data integrity before proceeding to the modeling phase. And we could solve the problem with those skills.

The second one is writing SQL queries. Writing efficient SQL queries for complex operations, such as aggregating IMDb ratings by genre or calculating directors' average ratings were the problems. But by breaking down complex queries into smaller parts and testing them incrementally, we could solve the problems.

**7 Conclusion and Future Improvements**

In this project, we developed a comprehensive movie database to help users easily find films matching their preferences by integrating detailed information on actors, directors, genres, ratings, and box office performance. Through meticulous data cleaning and modeling, we ensured data integrity and created a robust relational schema based on an E-R model. Despite challenges in dataset integration and SQL query optimization, we successfully overcame them by applying systematic techniques and iterative testing. This database offers valuable insights into the movie industry and serves as a foundation for further analysis and user-driven recommendations.

**ACKNOWLEDGEMENTS**

We would like to thank Professor John K. Leung from the Computational and Data Sciences Department at George Mason University for his valuable guidance in the CDS 302 course on Scientific Data and Databases. This project was a team effort, with Junghoon Choi handling data discovery and schema development, Heewon Yang entity relationship modeling, refining the schema, and creating the database, and Heeseung Moon and Wonbin Shin contributing to SQL research and key conclusions. Using SQLite3, we ensured smooth database management and ACID compliance, and we are grateful for the teamwork that made this project a success.

**REFERENCES**

Rehman, F. (2024, March 27). *IMDb movie reviews genres description and emotions*. Kaggle. https://www.kaggle.com/datasets/fahadrehman07/movie-reviews-and-emotion-dataset

G, R. A. (2022, October 29). *IMDB 5000+ movies & multiple genres dataset*. Kaggle. https://www.kaggle.com/datasets/rakkesharv/imdb-5000-movies-multiple-genres-dataset

anotherBadCode, anotherBadCode. (2022, May 15). *Boxofficecollections*. Kaggle. https://www.kaggle.com/datasets/anotherbadcode/boxofficecollections

Grijalva, D. (2021, July 23). *Movie Industry*. Kaggle. https://www.kaggle.com/datasets/danielgrijalvas/movies

Kanawattanachai, P. (2022, February 7). *Top 100 rotten tomatoes movies by genres*. Kaggle. https://www.kaggle.com/datasets/prasertk/top-100-rotten-tomatoes-movies-by-genres

Jadhav, R. (2024, April 27). *IMDb actors and movies*. Kaggle. https://www.kaggle.com/datasets/rishabjadhav/imdb-actors-and-movies/data