MOVIE INFORMATION DATABASE

*

Bilvani Veparala
Department of Computer Science
Georgia State University
Atlanta,GA,USA
bveparala1@student.gsu.edu

Rohith Reddy Kasarla
Department of Computer Science
Georgia State University
Atlanta,GA,USA
rkasarla1@student.gsu.edu

Vinay Kumar Revanuru
Department of Computer Science
Georgia State University
Atlanta,GA,USA
vrevanuru1@student.gsu.edu

***Abstract*—This report contains information about various aspects of movies such as cast, ratings, the tags assigned by the user to the movie, and the relevance of each tag to the actual movie.**

I. Introduction

The Internet Movie Database Popularly known as **IMDB**, is an online website that contains information about Movies, TV Shows, Video games, and online streaming content. The Information Includes cast, production house, ratings, Year of release, and so on. Over the years, this website has become a one-stop destination for movie and digital content lovers to find more information about their favorite movies and stars. Netflix is a popular streaming production service with an extensive library of collections of movies and Tv shows that air worldwide in various languages. It has revolutionized the way how content is consumed by the consumer and played a significant role in transforming audiences from choosing cable Tv and Movie Theatres to watching on the web browser with the help of the Internet

II. User Ratings, Tags and Their importance

As the number of movies and Tv shows coming out each year is increasing manifold, we have reached to a point where there are too many titles available but no increase in free time to watch them. Hence, the additional Information about the titles has become crucial for the production houses and consumers. Streaming services are in great need of feedback on the content that is on their website and how their audience is receiving it. This data is used in finding the patterns which help in better suggestions that help in enriching user experience and hence great subscription numbers and retention numbers for the Streaming giantsTags are one or two-word comments that are given to the movie by the movie watchers. These tags play a vital role in developing movie suggestions or recommendation systems. This helps streaming services in recommending movies, not only from their repository but also gives them valuable information on whether they have to acquire, renew or release a particular movie from their repository.

III. Description of Database

To create a Database system of movies and TV series, we have considered the Netflix Movies and TV shows database available on the Kaggle includes [**?**]. This dataset contains three CSV files, IMDb movies.csv, IMDb ratings.csv, and Netflix dataset.csv IMDB movies.csv contains descriptions of about eighty-two thousand movies and TV series with Information spanned in twenty-one columns, similarly IMDb ratings CSV has ratings of these movies with respect to people of various age ranges and countries in forty-nine columns. The Oscars dataset contains data of all the nominations for oscar awards since its inception. The Movie tag dataset contains data of tags that users have assigned to various movies and their relevance to the movie. The reviews dataset has 320,000 movie reviews by various users.

IV. Tools and Software used

PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. It was initially named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California. We are using this DBMS to create and manage our databases. Initial assessment of the data set is done in Spreadsheets.

V. Entity Relation Diagram

The ER diagram shown in fig1 gives the main idea of how our data will be organized into tables and how the tables will communicate with each other. our design has ten entity sets (tables), namely

- imdb_movies
- person_details
- oscar_awards
- listed_in
- netflix
- reviews
- imdb_tmdb_connector
- movie_tag_scores
- tag
- usertag

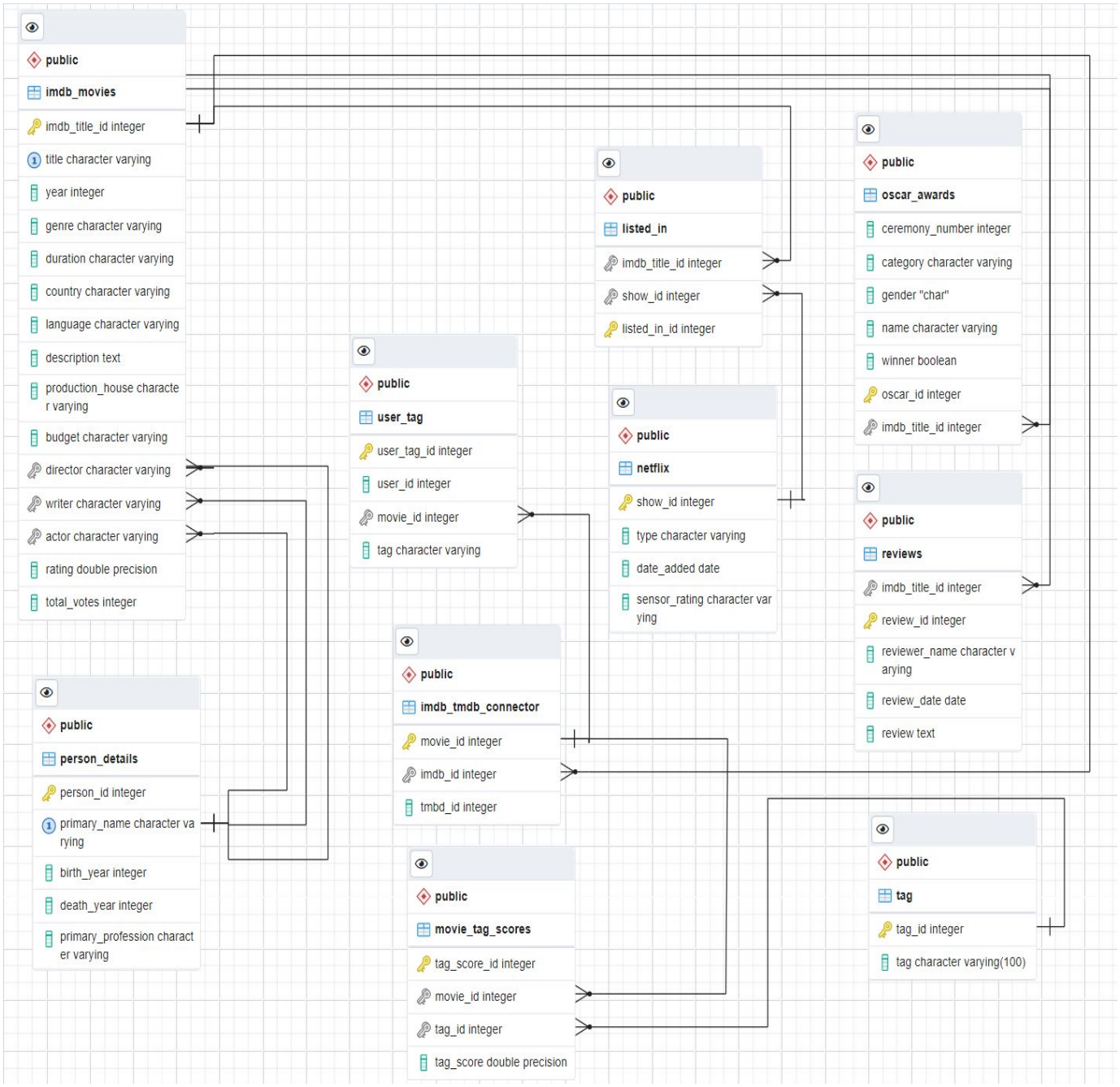

Fig. 1. ER diagram

VI. Description of Tables

.

IMDB_Movies

This is a table consisting of data that includes all the movies listed in IMDB with columns,

- 'imdb_title_id' – an auto-incrementing unique identifier used to identify the movies.
- 'title' – the title of the movie
- 'year' – the year of the movie release
- 'genre' – comma separated values of genres of the movie
- 'duration' – duration of the movie
- 'country' – comma separated values of movie origin country
- 'language' – comma separated values of languages the movie made in
- 'Description' – description about movie
- 'production_company' – Name of the production company.
- 'budget' – Budget allocated for each movie.
- 'director' – Director's name of the movie, can join with

'person_details' and get details about the director.
- 'writer' – The writer's name of the movie, can join with 'person_details' and get details about the writer.
- 'actor' – Main actor name of the movie, can join with 'person_details' and get details about actor.
- 'rating' – Average of the total votes.
- 'total_votes' – total count of votes

Netflix This table consists of all the movies and TV Shows listed in Netflix

- 'show_id' – Auto incrementing unique identifier used to identify each movie/Tv show listed in Netflix.
- 'type' – Type of the content (Movie/TV Show).
- 'date_added' – Date the movie/tv show added to Netflix. 'sensor_rating' – sensor rating of the movie/Tv show.

Listed_in Lists the movies of 'Netflix' which are listed in 'imdb_movies' table. As there is no link between imdb_movies and Netflix, we can only link them using this table which maps each 'show_id' to its respective 'imdb_title_id'.

- listed_in_id' – unique identifier to identify each listing.
- 'imdb_title_id' – referenced from 'imdb_movies' table.
- 'show_id' – referenced from 'Netflix' table

Person_details Some information about a few individuals working in the film field. Includes their ID which is a unique 'Imdb' person identifier that can be further used to link this dataset possibly with other datasets.

- 'person_id' – Unique person identifier given by IMDB.
- 'primary_name' – the name of the individual.
- 'birth_year' – the birth year of the individual.
- 'death_year' – death year of the individual.
- 'primary_profession' – comma-separated values of the profession of the individual.

Oscar_awards This table contains information about Oscar awards given to films for different categories. We can link the title and year from the 'imdb_movies' table with this table to get information on the awards of the film individually and we can also join them using the names of 'director or writer or actor' to see if they got any award from Oscars.

- 'oscar_id' – a unique identifier used to identify each entry of the table.
- 'ceremony_number' – the Oscar ceremony number
- 'category' – the nomination category of the movie.
- 'gender' – gender of the individual nominated.
- 'name' – the name of the individual nominated
- 'winner' – T (if nomination won) or F (if nomination lost)
- 'imdb_title_id' – referenced from 'imdb_movies' table.

Reviews The reviews dataset has 320,000 movie reviews by various users.

- 'imdb_title_id' – referenced from 'imdb_movies' table.
- 'review_id' – unique identifier to identify each review given by the reviewer.
- 'reviewer_name' – the name of the reviewer.
- 'review_date' – date of the review.
- 'review' – review content.
- 'no_of_reviews' – number of reviews for the film.

Link the table which links the imdb_movies with ratings and movies from other sites like 'TMDB'.

- 'movie_id' – unique identifier for a movie from another site.
- 'imdb_id' – IMDB unique movie identifier.
- 'tmbd_id' – TMDB unique movie identifier.

Movie_tag_scores This table contains the information about tag scores of each unique tag given to a movie.

- 'movie_tag_id' - a unique identifier used to identify each entry of the table.
- 'movie_id' – movie identifier referenced from 'link' table.
- 'tag' – the tag given to the movie by the user.
- 'tag_score' – score given to the tag assigned to the movie based on its relevance with the movie.

Tag This table contains information about the tag and their IDs.

- 'tag_id' – unique identifier of the tag.
- 'tag' – name of the tag.

User_tag This table contains information about the tags assigned to the movie by the users.

- 'user_tag_id' - a unique identifier used to identify each entry of the table.
- 'user_id' – a unique identifier used to identify the user.
- 'movie_id' - movie identifier referenced from 'link' table.
- 'tag' – name of the tag given to the movie by the user.

VII. Normalization of datasets

Our project idea was to normalize the data as much as possible while ensuring that the database handles null values efficiently. When we talk about handling null values, the best way is to make sure we don't have any null values. Still, in a real-world scenario, that is next to impossible, but when designing a database, we can minimize the chances of getting null values. We did the same with our database, for example, The priority of the "Listed_in" table is to map the "imdb_title_id" of "Imdb_movies" with the "show_id" of Netflix. We could have aggregated Netflix columns with Imdb _movies. We could have got rid of both the "Netflix" and "Listed_in" tables, but that will would result in unwanted null values in the "Imdb_movies" table because "Imdb_movies" consists of 85855 rows of data. In contrast, Netflix only contains 7786 rows of data. In an ideal situation, all three columns on "Netflix", added to "Imdb_movies" would have 78069 null values, which means 78069 null values are repeated for all three columns (78069*3). With this normalization, we are atleast eliminating 234207 (78069*3) null values. We followed the same normalization process during the creation of each table and made sure that null values are minimized and improved the efficiency of the database.

Normalization also helps to improve data integrity, which refers to the accuracy, completeness, and consistency of the data in the database. By dividing data into smaller, more

Table Name	Number of rows of data
Imdb_movies	85855
Imdb_tmdb_connector	8562
Listed_in	2688
Movie_tag_scores	31000
Netflix	7786
Oscar_awards	8912
Person_details	49045
Reviews	50000
Tag	1128
User_tag	3479

Fig. 2. cardinality of tables

focused tables, and creating relationships between these tables, you can ensure that data is entered and stored in a consistent and accurate way.

Normalization can also improve the performance of the database, as it can reduce the amount of data that needs to be accessed to retrieve a specific piece of information. This can help to speed up queries and improve the overall performance of the database.

The above table shows the number of rows of each schema in our database

VIII. Implementation and working

The data we obtained from the internet was unclean and flat. It also contained many redundant columns and repeating data. To normalize the data, We initially loaded these unnormalized datasets into our pgAdmin database to analyze the columns. We made the IMDB_Movies dataset as our core dataset. We modified the remaining data accordingly by keeping this Dataset in the spotlight. This process helped eliminate numerous columns, particularly in the movie tags and reviews Dataset, that aid in improving the performance while joining these tables when writing queries.

After normalizing the data with the help of PgAdmin, we finally loaded the data into a new database with the columns having descriptions as mentioned above. The data loaded perfectly and it was ready to deploy. When we created an ER diagram from the tool, we got a consistent diagram as shown in the figure, with all the tables perfectly connected among each other just as we designed. The count of unique values of primary keys corresponded with the total number of rows in the table, which is consistent with the definition of the primary key. Our database system requires atmost five joins to traverse from one relational schema to another. and it occurs when we try to join user_tag table with the Netflix table

It really amazed us that the code took much longer runtimes when it came to counting the number of unique rows of a query compared to just doing that query. When we researched it, we found out that this is due to the larger runtimes to traverse through the B+ Trees structure

When we listed the number of movies released each year, we observed that the number of movies released in each year has increased or there was a slight decrease compared to previous years. but there was a sharp decline in the number of movies in the year 1943, compared to 1942 and the trend continued in 1944 and 1945 it reached rock bottom, it had fewer movies than in 1931, all this can be accounted to world war 2.

When we were analyzing the trend of popular genres over the years, the Drama genre unanimously stood at the first position for the major share of years from 1900 to 2020 except in the decade of 1980s when comedy movies started ruling the box office, It's more accurate to say that the 80s saw a rise in the production of comedy movies, which were popular with audiences at the time. This increase in the popularity of comedies may have been due to a number of factors, including the economic growth and social change of the era, and the emergence of talented comedians.

IX. Challenges

Once the database schema has been designed, the next challenge is to populate the database with data. This can be a labor-intensive process, especially if you are working with a large amount of data. You will need to ensure that the data is entered accurately and consistently and that any errors or inconsistencies are corrected.

Every time we modify our database during the development process, The ER Diagram generation through PGAdmin used to be challenging as ERD generated was not properly aligned and hard to understand, and after every modification, we had to reorganize the whole ERD to make it easier to understand.

PG Admin GUI did not have the functionality to change the datatype of an attribute from int to char once the table is created, it only allowed to change from char to varchar, text, or int to bigint instances, this made us delete the entire table and again create the new table from scratch

Another challenge is the Format compatibility of the dataset with pgAdmin. pgAdmin only supports certain file formats, such as CSV and SQL, so we had to convert our dataset to one of these formats before uploading it, and during this conversion to CSV, we encountered multiple string format errors which we removed by using pandas in python

Overall, it is important to carefully prepare the dataset and ensure that it is compatible with pgAdmin before attempting to upload it. This can help to avoid any potential issues and ensure that the analysis is accurate and reliable

As we made IMDB_movies our main dataset, we had to reinsert the remaining table's data to maintain the foreign key constraint, This step significantly improved the performance as we got rid of many redundant and unnecessary tuples of data.

X. conclusion

Once established, this database will be pretty helpful for Movie Recommendations to individual users. The insights gained through these ratings would also help the production

houses on making a decision on the kind of movies that are adored around the world, allowing them to plan their future films carefully. We made this Dataset further accessible to other movie database services such as TMDB. we could extend this database further. we can also add the movies that are listed in various streaming services making this database a one-stop destination to navigate movies similar to justwatch.com and also add other features such as our own movie recommendation system from all the sources based on the user's watch list and preferences, reviews of the movies.

XI. References

- The Entity-Relationship Model - Toward a Unified View of Data,Peter-Pin,Shan-Chen, https://dl.acm.org/doi/pdf/10.1145/320434.320440
- The Postgre SQL Documentation, https://www.postgresql.org/docs/current/app-psql.html
- IMDB Movies ,Netflix Datasets, https://www.kaggle.com/datasets/mirajshah07/netflix-dataset?
- Oscar Awards , Academy awards dataset oscars https://www.kaggle.com/datasets/dharmikdonga/academy-awards-dataset-oscars
- Movie Tag dataset, https://www.kaggle.com/datasets/aigamer/movie-lens-dataset?
- Person_details from IMDB Dataset, https://www.kaggle.com/datasets/komalkhetlani/imdb-dataset?
- 32000 movie review sentiment analysis, https://www.kaggle.com/datasets/nikosfragkis/imdb-320000-movie-reviews-sentiment-analysis