PROJECT REPORT

Bansi Shah(bks7385)
Parth Bhardwaj(pb2640)

PROJECT DESCRIPTION

Note: The port number used to deploy our project is 8604

Our project “The Movie Database Project” aims to store data related to movies in an efficient manner following the principles of database systems and generate insights around movies that a user of our database or perhaps any cinephile might find interesting.

Our application can be thought of as goodreads for movies, where a registered user can view information about movies such as the movie Title, runtime, view lists that other users have made around movies, read reviews about a movie, get information about the trailer of a movie etc. This user can use our application to generate important insights around the data that we have besides viewing different data points.

Background : we have collected our data from TMDB for Movie metadata and Youtube for trailer related data by making api calls, we used python to process all the data and collate it in different csv files

Data : We have a total of 8 tables and the description of our data is given below

Table Name	Number of records	Description
Movies	10812	The table has information about the movie title, release date, runtime, status(eg: released, in production, etc), language it’s made in, popularity and whether the movie is adult or not. It also had the ids of the genre, spoken language and production country which refer to their respective table for more information.

Users	79	Contains username and their emails
Trailer	837	Contains runtime, trailer title and trailer statistics for movies, eg: number of views, likes and comments on the trailer.
Reviews	328	Contains reviews for a movie by a user
List	63	Contains the lists made by a user, for example a user “max” can have a list named “best horror” and have n number of movies in it
Production Countries	164	It stores the ids of different countries to refer to movies released there and the name of the country associated with the id.
Spoken Languages	131	Contains the language spoken in the movie besides the language the movie was originally made in (i.e “original_language”
Genre	19	Contains Genre id and name, eg: “Comedy”

INCORPORATING FEEDBACK

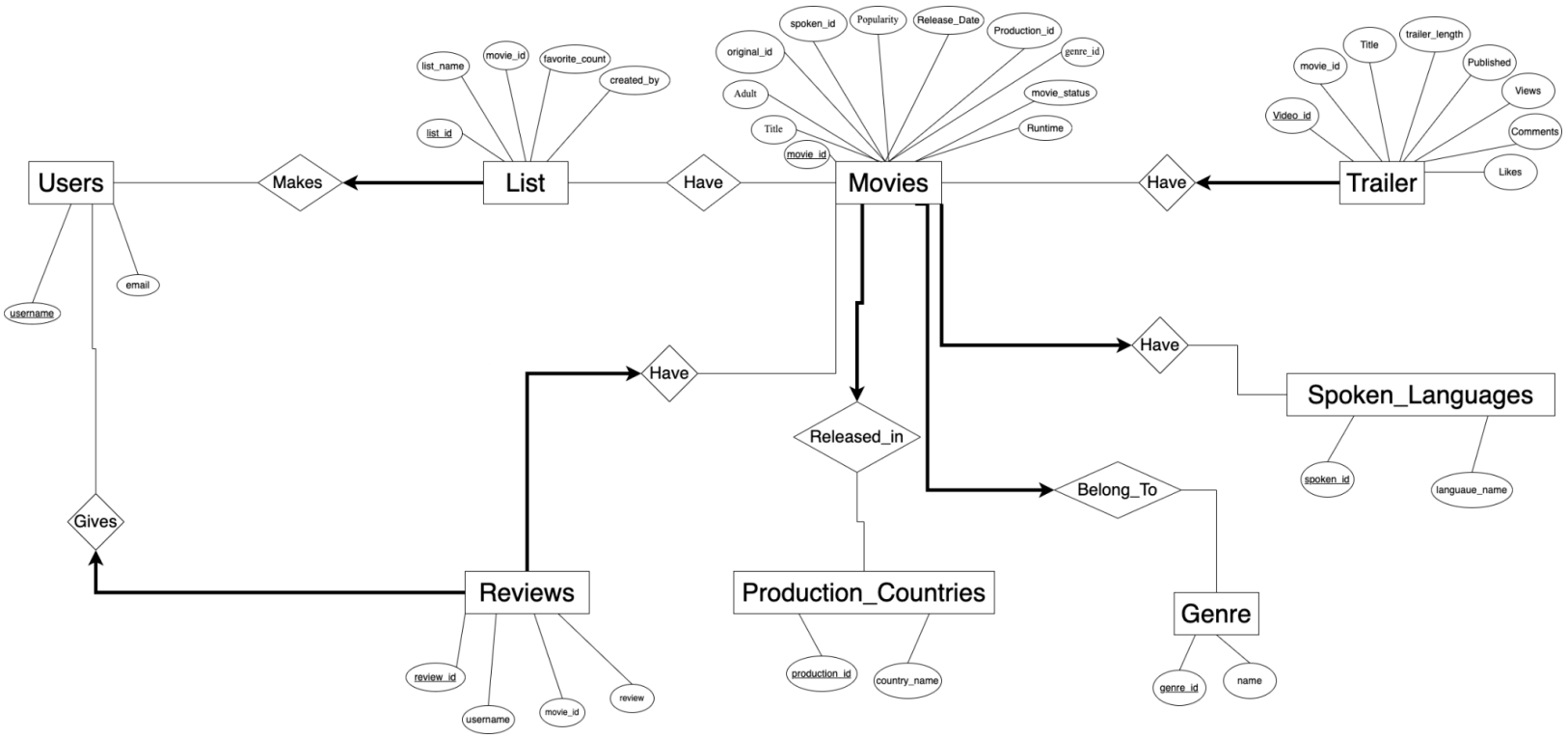
We carefully analyzed the feedback given and made the changes. We have included the changed ER diagram and schema below. We have also provided the schema file which includes the insert statements to insert data in the tables.

Business Rules

- User :
  - A user may write 0, 1 or multiple reviews
  - A user may make 0, 1 or multiple lists
- List:
  - Every list must be created by exactly 1 user
  - Every list may contain 0, 1 or multiple movies
- Review:
  - Each review is given by exactly 1 user
  - Each review is about exactly 1 movie
- Movie:
  - A movie must be released in exactly 1 country

    - A movie must belong to exactly 1 genre
    - A movie must have exactly 1 spoken language
    - A movie may have 0, 1 or multiple reviews
    - A movie may be in 0, 1 or multiple lists
    - A movie may have 0, 1 or multiple trailers
- Production_Countries:
    - A production country may have 0, 1 or multiple movies released there
- Genre:
    - A genre may have 0, 1 or multiple movies belonging to it
- Spoken_Languages:
    - A spoken language may be in 0, 1 or multiple movies
- Trailer:
    - A trailer belongs to exactly 1 movie

ER Diagram



Entity Sets:

1. Users- Username, Email,
2. Movies - Movie_Id, Title, Adult, Original_Language, Spoken_Id, Popularity, Release_Date, Production_Id, Runtime, Genre_Id, Movie_Status
3. List - List_Id, List_Name, Movie_Id, Created_By, Favorite_count
4. Reviews - Review_Id, Created_By, Movie_Id, Review

5. Trailer - Video_Id, Movie_Id, Title, Trailer_Length, Published, Views, Likes, Comments
6. Genre - Genre_Id, Name
7. Spoken_Languages - Spoken_Id, Language_Name
8. Production_Countries - Production_Id, Country_Name

Relationship Sets:

1. Makes - Relationship between Users and List
2. Gives - Relationship between Users and Reviews
3. Have - Relationship between List and Movies
4. Have - Relationship between Movies and Reviews
5. Have - Relationship between Movies and Trailer
6. Have - Relationship between Movies and Spoken_Languages
7. Released_in - Relationship between Movies and Production_Countries
8. Belong_To - Relationship between Movies and Genre

Schema

```
DROP TABLE IF EXISTS list;
DROP TABLE IF EXISTS trailer;
DROP TABLE IF EXISTS reviews;
DROP TABLE IF EXISTS movies;
DROP TABLE IF EXISTS genre;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS production_countries;
DROP TABLE IF EXISTS spoken_languages;
CREATE TABLE genre(
   genre_id INTEGER  PRIMARY KEY
  ,name     VARCHAR(100) NOT NULL
);


CREATE TABLE production_countries(
   production_id VARCHAR(2) PRIMARY KEY
  ,country_name          VARCHAR(32) NOT NULL
);


CREATE TABLE spoken_languages(
   spoken_id     VARCHAR(2) PRIMARY KEY
  ,language_name VARCHAR(23) NOT NULL
);
```

```sql
CREATE TABLE users(
   username VARCHAR(100) PRIMARY KEY
  ,email    VARCHAR(100) NOT NULL
);



CREATE TABLE movies(
   adult                BOOLEAN
  ,genre_id             INTEGER NOT NULL references genre(genre_id)
        ,movie_id             INTEGER  PRIMARY KEY
  ,original_language    VARCHAR(2)
  ,popularity           NUMERIC(8,3)
  ,production_id VARCHAR(2) NOT NULL references
production_countries(production_id)

  ,release_date         DATE
  ,runtime              INTEGER
,spoken_id     VARCHAR(2) NOT NULL references spoken_languages(spoken_id)
  ,movie_status               VARCHAR(15)
  ,title                VARCHAR(9999)


);

CREATE TABLE trailer(
   Video_ID  VARCHAR(100) PRIMARY KEY
  ,movie_id  INT  NOT NULL references Movies(movie_id)
  ,Title     VARCHAR(100)
  ,trailer_length    VARCHAR(7)
  ,Published DATE
  ,Views     INTEGER
  ,Likes     INTEGER
  ,Comments  INTEGER
--    ,movie_id FOREIGN KEY references Movies(movie_id)
);

DROP TABLE IF EXISTS reviews;
CREATE TABLE reviews(
```

```
   review_id VARCHAR(100) NOT NULL PRIMARY KEY
  ,username  VARCHAR(100) NOT NULL references Users(username)
  ,movie_id  INTEGER  NOT NULL references Movies(movie_id)
  ,review    VARCHAR(9999) NOT NULL

);


CREATE TABLE list(
   list_id        SERIAL  PRIMARY KEY
  ,list_name      VARCHAR(100)
  ,movie_id      INTEGER references Movies(movie_id)
  ,favorite_count INTEGER  DEFAULT 0
  ,created_by     VARCHAR(100) NOT NULL references Users(username)


);
```

DATA LOADING PROCEDURE

- For data loading we have included the insert statements in the “load.sql” file in the data folder, executing this file will create the tables and populate them
- Procedure:
    - Now that we have the data files on our local system we transferred the schema.sql file and the load.sql file to jedi using ssh
    - After going into psql we execute the sql file using e.g $ \i schema.sql
    - Then we execute the load.sql file in the same manner
    - Note: while executing schema.sql some insert commands have conflicting values that have been included and not removed intentionally.
        i. Example: in the Trailers table the comments attribute is of type int but some insert commands have disabled as the value. This is because when we use the Youtube API to get the statistics of a

trailer some videos don't share these numbers and removing these statements is a tedious process. Hence we included this, in fact it's a good Quality check for us.

ii. Some comments have characters for which encoding "WIN1252" has no equivalent in encoding "UTF8" hence such comments do not get included in our database.