DBMS - II

Assignment -2

TEAM NO. 17

Aim:
Build an efficient database and manage it effectively from IMDB datasets in PostgreSQL

Team Members:

Raviteja Namani - CS18BTECH11032
Geethika Sowmya - ES18BTECH11025
Katravath Rajesh - CS18BTECH11023
Mathangi Jedidiah - CS18BTECH11028

Requirements:

1) PostgreSQL >=9.6
2) Python3 and a package installer for python3.
3) CSV and JSON modules installed in python3.
4) TMDBSimple Module
5) API Keys for TheMovieDB (for release dates) and RapidAPI (for awards)

Introduction

A DBMS supplies a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have correlated meanings.

Database Description:

1. originalTitles - This orginalTitles table contains all the original titles and information corresponding to each of them.

```
CREATE TABLE originalTitles(
   tconst TEXT PRIMARY KEY,
   titleType TEXT,
   primaryTitle TEXT,
   originalTitle TEXT,
   isAdult TEXT,
   startYear TEXT,
   endYear TEXT,
   runTime TEXT,
   genres TEXT );
```

FIELD NAME	DATA TYPE	CONSTRAINT	DESCRIPTION
tconst	TEXT	PRIMARY KEY	unique id
titleType	TEXT	-	TV series/movie
primaryTitle	TEXT	-	first title
originalTitle	TEXT	-	original title
isAdult	TEXT	-	rated adult
startYear	TEXT	-	starting year
endYear	TEXT	-	ending year
runTime	TEXT	-	run time of the film
genres	TEXT	-	genre title falls into

2. title_genres - Each movie can have multiple genres and vice versa. So create a separate table for it because it is a many-many relationship.

```
create table title_genres(
   tconst TEXT,
   genre TEXT,
   PRIMARY key (tconst,genre),
   CONSTRAINT tconst_fk FOREIGN key(tconst) references originalTitles(tconst) on
delete cascade
);
```

FIELD NAME	DATA TYPE	CONSTRAINT	DESCRIPTION
tconst	TEXT	tconst_fk FOREIGN key(tconst) references originalTitles(tconst) on delete cascade	
genre	TEXT	- PRIMARY key (tconst,genre)	genre

3.all_titles - Table contains all titles in different languages and regions
.

```
CREATE TABLE all_titles(

   titleid TEXT ,
   ordering TEXT ,
   title TEXT,
   region TEXT,
   title_language TEXT,
   types TEXT,
   attributes TEXT,
   isOriginalTitle TEXT,
   primary key(titleid,ordering)
   );
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
titleid	TEXT	-
ordering	TEXT	-
title	TEXT	-
region	TEXT	-
title_language	TEXT	-
types	TEXT	-

attributes	TEXT	-
isOriginalTitle	TEXT	-
		primary key(titleid,ordering)

4. cast_and_crew -This table contains all the cast and crew members.

```
CREATE TABLE cast_and_crew(
   nconst TEXT PRIMARY KEY,
   primaryName TEXT,
   birthYear VARCHAR(10),
   deathYear VARCHAR(10),
   primaryProfession TEXT ,
   knownForTitles TEXT
   );
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
nconst	TEXT	PRIMARY KEY,
primaryName	TEXT	-
birthYear	VARCHAR(10)	-
deathYear	VARCHAR(10)	-
primaryProfession	TEXT	-
knownForTitles	TEXT	-

5. CastAndCrewProfession -The table contains professions of each member from cast_and_crew. References to cast number id.

```
CREATE TABLE castAndCrewProfession(
   nconst TEXT ,
   profession TEXT ,
   PRIMARY KEY(nconst,profession),
   CONSTRAINT fk_nconst FOREIGN KEY(nconst) REFERENCES
cast_and_crew(nconst)
);
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
nconst	TEXT	fk_nconst FOREIGN KEY(nconst) REFERENCES cast_and_crew(nconst)
profession	TEXT	- PRIMARY KEY(nconst,profession)

6.Table episodes - a temporary table to copy information from title.episodes.tsv and then update in originalTitles.

```
create temporary table episodes(
   tconst TEXT,
   parentTconst TEXT,
   seasonNumber TEXT,
   episodeNumber TEXT
);
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
tconst	TEXT	-
parentTconst	TEXT	-
seasonNumber	TEXT	-
episodeNumber	TEXT	-

7.Table castAndCrewTitles - contains the titles of each member. References to person id from cast_and_Crew table.

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT

nconst	TEXT	fk_nconst FOREIGN KEY(nconst) REFERENCES cast_and_crew(nconst) ON DELETE cascade
titleid	TEXT	-
		PRIMARY KEY(nconst,titleid)

```
CREATE TABLE castAndCrewTitles(
   nconst TEXT ,
   titleid TEXT ,
   PRIMARY KEY(nconst,titleid),
   CONSTRAINT fk_nconst FOREIGN KEY(nconst) REFERENCES
cast_and_crew(nconst) ON   DELETE cascade
);
```

8. Table ratings - a temporary table too having information about ratings of titles.

```
create temporary table ratings(
   tconst TEXT,
   rating TEXT,
   votes TEXT
   );
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
tconst	TEXT	-
rating	TEXT	-
votes	TEXT	-

9.Table principals - contains principal members for titles and their information in title like character name etc

```
create table principals(
   tconst text,
   ordering text,
   nconst text,
   category text,
   job text,
   characters text
);
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
tconst	TEXT	-
ordering	TEXT	-
nconst	TEXT	-
category	TEXT	-
job	TEXT	-
characters	TEXT	-

10.Table crew - This is a temporary table too to copy the from tsv

```
create temporary table crew(
    tconst TEXT,
    directors TEXT,
    writers TEXT
);
```

FIELD NAME/ DESCRIPTION	DATA TYPE	CONSTRAINT
tconst	TEXT	-
directors	TEXT	-
writers	TEXT	-

11.Table directors - A separate table for directors. referenced to originaltitles by tconst

```
create table directors(
    tconst TEXT,
    director TEXT,
     PRIMARY key(tconst,director),
    constraint tconst_fk foreign key (tconst) references originalTitles(tconst)
);
```

FIELD NAME	DATA TYPE	CONSTRAINT
tconst	TEXT	tconst_fk foreign key (tconst) references originalTitles(tconst)
director	TEXT	-
		PRIMARY key(tconst,director)

12.Table writers - A separate table for writers. referenced to originaltitles by tconst

```
 create table writers(
    tconst TEXT,
    writer TEXT,
    PRIMARY key(tconst,writer),
    constraint tconst_fk foreign key (tconst) references originalTitles(tconst)
);
```

FIELD NAME	DATA TYPE	CONSTRAINT
tconst	TEXT	tconst_fk foreign key (tconst) references originalTitles(tconst)
writer	TEXT	
		PRIMARY key(tconst,writer)

13.Temporary table releasedates- This is an extra table that has been crawled from https://www.themoviedb.org/documentation/api.

```
create temporary table releasedates(
    tconst TEXT,
    title_type TEXT,
 releasedate TEXT
);
```

FIELD NAME	DATA TYPE	CONSTRAINT	DESCRIPTION
tconst	TEXT	-	
title_type	TEXT	-	type ot the title
releasedate	TEXT	-	release date

14. **Table awardsAndNominations** - contains information about the awards and nominations the cast member has received. And if they have won the award. The nconst is referenced to cast_and_crew

```
create table awardsAndNominations(
    nconst TEXT,
    awardname TEXT,
    category TEXT,
    isWinner TEXT,
    constraint nconst_fk foreign key (nconst) references cast_and_crew(nconst)
);
```

FIELD NAME	DATA TYPE	CONSTRAINT	DESCRIPTION
nconst	TEXT	nconst_fk foreign key (nconst) references cast_and_crew(nconst)	
awardname	TEXT	-	
category	TEXT	-	
isWinner	TEXT	-	

Changes made from the ER Diagram submitted:

1) In the ER Diagram submitted in the previous assignment, We have maintained different tables for movies, TV Series and Episodes. But in this assignment, we have given everything in the same table: originalTitles. This is because we had assumed that different TSV files would be provided but that is not the case here.
2) We had only maintained awards for cast and crew members but not for movies/TV Series'. Because crawling the data took so long and we had expired the maximum number of API requests.
3) Locations and production company tables have not been implemented because of the lack of datasets.
4) Most of the columns which have been assumed compulsory in ER Diagram turned out to be optional because the datasets have not all fields filled for every row.

Logical Design:

Our entire SQL file is written in postgreSQL for building a database for the Imdb website.On analysis of relatively raw data obtained from given tsv files ,We found that the datasets are very corrupted and have a lot of unnecessary data.Some files have ids for titles and persons whose ids have not been assigned too. This document assumes those are corrupted data and have been deleted before referencing between the tables.

We have created corresponding tables to contain relevant data . Everything is put under text datatype to make sure no error is encountered while copying from a tsv file and altered accordingly for example to int or bool as we built the relations.
Each movie can have multiple genres and vice versa. So created a separate table for it because it is a many-many relationship.

We also smoothed out the data from comma separated string to an array then Used the unnest function to obtain each entry and insert into the specific table with tconst and genre as columns, Deleted the default first entry which has column names. And also corrupt or repeated columns that are already taken care of or not necessary.

Calculated age from birth and death year for cast and crew_table
1) if birth year not known, put NULL,
2) If the death year is not known, the person is still alive. So calculate from the present date.
3) or calculate death year-birth year
to get better and more specific querying results.

Updated the tables with precise data

For eg, Updated the values from episodes table to originaltitles table by matching tconst
Create temporary tables to extract raw data and link the data to the existing tables
Insert the values from temporary table to directors by unnest and string_to_array
Via primary keys or constraints.

At a basic level our modeling of the database has been to extract the whole raw data without missing out, smooth out data types and split information that can be, into separate columns .
Create separate tables for all the given information and organise it to get precise queries.
Edit the tables with unwanted extra data and link all the tables with primary keys and constraints to answer complex queries.

Querying the database :

Users of a database could look for :
1. Given a specific time period, a user might be interested in finding the top n box office movies aired in a given region
2 Given an actor's/actress' name, a user might want to find the movies that the actor/actress acts in.
3.Given the title of the movie ,a user might want to find out who are the cast members for the movie specified and what are the awards won by a given movie.
4.A user might want to search all the awards won by a given movie. Or the career of the actor spanning over years

Contributors of the database:

1. A contributor might want insert or delete a movie or a show
2.A contributor might want to insert or update the database with any other relevant information to the existing database.

Extra Datasets for Bonus:

Crawled Data for releasing dates:
This API has been used for obtaining data regarding release and AIR dates of movies, TV Series and episodes. A python script has been written which uses the module "tmdbsimple" for sending API requests to the URL and obtains the response in the form of JSON. The python script reads the "title.basics.tsv" to abstract titleids of all titles and does API requests using the tmdbsimple module. A csv file is created in write mode, the

JSON data is converted into CSV format and writer. The problem here is that there are 10 million titles and doing one API request at a time, it takes 11 days for whole data to get extracted. So, only limited data has been attached with this just for reference.

Crawled Data for Awards:
This API is used for obtaining data regarding awards received by cast members. Unfortunately, it allows only 500 API requests on free trial. So, only limited data has been used for this. The same procedure is followed here too with python script. The python scripts for both of them have been attached with this report. Change the API key and generate the datasets to run.