Assignment - 2 : CS3563 : Introduction to DBMS - II

Harsh Agarwal `cs15btech11018` | Vishwak Srinivasan `cs15btech11043` | Sahil Yerawar `cs15btech11044`

1 Introduction

The objective of this assignment was to create and populate relational tables based on the entity relationship diagram (ER diagram) in the previous assignment.

The technology used was PostgreSQL[1]. Additionally, we made use of the TMDB[2] to help extract data that was unavailable to us through the IMDB dataset. We have also made use of the Oscar Dataset. from which we managed to get certain award information, that could fit within the scope of the specifications dictated by the designed ER diagram.

2 Statistics of Data available on the Database

In total, the IMDB dataset comprised of 3793252 titles which were either a movie or TV Series[3]. This is marked by a `titleType` attribute in the IMDB file named `title.basics.tsv`. In our case database, a "Movie" in the database corresponds to just a `movie`, whereas a "TV Series" corresponds to `tvSeries` or `tvEpisode` or `tvMiniSeries`. Below are the sizes of the populated tables and the number of rows:

List of relations

Schema	Name	Type	Owner	Size	Description
public	awards	table	postgres	264 kB	
public	filming_location	table	postgres	4056 kB	
public	genres	table	postgres	259 MB	
public	languages	table	postgres	11 MB	
public	person	table	postgres	454 MB	
public	picture	table	postgres	409 MB	
public	production_company	table	postgres	6064 kB	
public	rating	table	postgres	30 MB	
public	release_location	table	postgres	89 MB	
public	role	table	postgres	1272 MB	

(10 rows)

Comments

Table	Children	Parents	Columns	Rows
awards		2	7	1,734
filming_location		1	2	75,660
genres		1	2	5,480,283
languages		1	2	217,860
person	2		4	8,435,455
picture	9	1	13	3,793,252
production_company		1	3	93,633
rating		1	3	607,078
release_location		1	2	1,685,882
role		2	3	21,494,867
10 Tables			**41**	**41,885,704**

[1] `https://www.postgresql.org/`

[2] `https://www.themoviedb.org/?language=en`

[3] as on January 25th, 2018

3 Parsing and Processing

3.1 Parsing

We first scanned the existing files in the IMDB datasets. We could get preliminary information for movies / TV series such as:

- Release Titles and Primary Titles (well-known title)
- Adult ratings
- Start Year (for TV Series, analogous to Release Year for Movies)
- End Year (applicable to TV Series)
- Genres
- Runtime (Duration)

For cast and crew, we could obtain:

- Movies associated with and their roles in those movies
- Name, birth year and death year

We were also able to obtain release locations and languages associated with the movie from the IMDB dataset as well.

3.2 Processing

Firstly, to easily distinguish between movies and TV series, we decided to include a `IsMovie` field in the table. As the name suggests, this field denotes if a given "Picture" is a Movie or TV Series. We continue to refer to both collectively as a "Picture"

Then, owing to the many-to-many nature between Pictures and genres, we decided to move it away from the master table. A similar approach was taken with respect to release locations and languages, which were not extremely specific hence inducing a many-to-many behaviour, and was moved away from the master.

We created new tables for peoples general attribute such as name, birth year and death year (if applicable), and a separate table linking Pictures and people with their roles. Due to the in-specificity of the role information in the IMDB dataset, we decided to categorize roles into 7:

Given roles	Categorized Role
Actor, Actress	**Actor**
Director	**Director**
Writer	**Writer**
Composer	**Music**
Producer	**Producer**
Cinematographer, Editor, Production Designer	**Crew**
Others (e.g., self)	**Miscellaneous**

For the release locations, we were given the locations in terms of the ISO-3166 codes. We used `iso3166`[4], a PyPI package for helping convert these to actual country names. Similarly, languages were given to us in terms of the ISO-639 codes, for which we used `iso639`[5], a PyPI package for helping convert these code to language names.

Most of our processing was done using Pandas, a well-known Python library for efficient database management. We convert the required tables into comma separated value format files, which we later feed into PostgreSQL.

4 Crawling and more Processing

4.1 Crawling

Since most of the fields required by the ER diagram were empty by default, we tried to scrap some information from the internet. Scraping was done in two ways:

- Use existing augmented datasets that can be ported with the existing tables built.
- You crawl what you want.

Consequently, we were able to obtain additional information to the Picture table which was originally not available to us such as Gross Box Office, Budget, Production Companies, Filming / Production Locations. We were also able to get some award information pertaining to these Pictures as well.

4.1.1 Picture information

The picture information was obtained from two sources. One source was a small partition of the TMDB dataset, which we found on Kaggle. Another source was through the API[6] calls made to TMDB, which gave us information of the same format as the data obtained from Kaggle.

4.1.2 Award information

We have used the Oscar dataset, which contains the list of all the awards and nominations from 1927 till 2015 in this case. We found this dataset on Kaggle[7]. Since there were various inconsistencies faced during obtaining information about the award recipients and picture to which the award belonged to, we decided to ignore some types of awards altogether like awards related to documentaries, production companies, music and so on. Using the existing information, we were able to link the awards with a movie and/or a person.

4.2 More Processing

Once, these information was obtained, we had to restructure the crawled data to be able to fit in with existing tables. New tables were constructed with production and filming information in them. However, it is not certain that *all* the pictures in our final database have this information. We also created an awards table for certain pictures and cast and crew members.

[4] `https://pypi.python.org/pypi/iso3166/0.6`
[5] `https://pypi.python.org/pypi/iso-639`
[6] `https://www.themoviedb.org/documentation/api`
[7] `https://www.kaggle.com/theacademy/academy-awards/data`

5 Assembling the CSVs to form the Relational Database System

After all the CSVs were assembled, we moved on to populating the database. First we created all the tables based on the description given in Section 6. Then the data was populated using PostgreSQL `COPY`[8] command.

',' was chosen as the delimiter, `\N` was used to denote NULL values and `unicode` encoding was used. All the PostgreSQL commands used in this assignment are included in the submission.

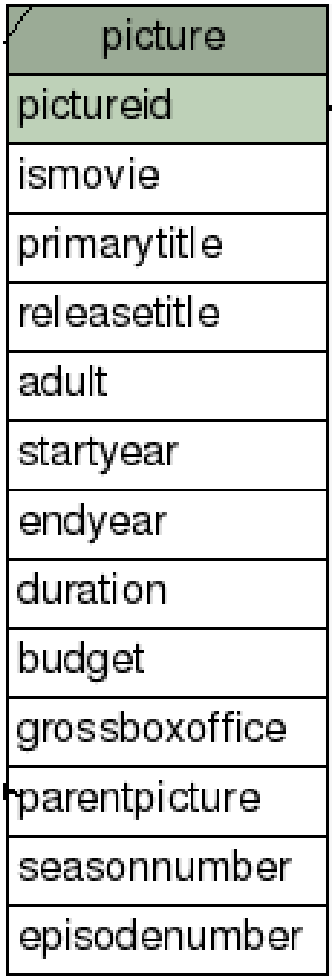
The link for the raw CSV files used to generate these tables can be found at this link.

6 Description for the Tables

6.1 `Picture`

The `picture` table consists of 13 attributes, which represent the essential details to characterize the Picture. Some explanation for each attribute is given below:

- `PictureID:` A unique ID for each Picture
- `IsMovie:` A boolean for differentiating a Movie and a TV Series
- `PrimaryTitle:` The Primary Title of the movie (well - known title)
- `ReleaseTitle:` The Release Title. IMDB classifies movies with same Primary Title but with different Release Title as different movies.
- `Adult:` A boolean marker for adult content in the movie (a form of universal certification)
- `StartYear:` This is the release year in the context of a movie and series start year in the context of a TV Series
- `EndYear:` This is the last airing year for a TV Series. Defaults to NULL for Movies.
- `Duration:` The duration of the movie / TV episode
- `Budget:` Amount of money take to film
- `GrossBoxOffice:` Gross earnings through ticket sales
- `ParentPicture:` A self-referential key which highlights the name of the TV Series. Note that the Primary title / Release title are for TV episodes.
- `Season Number:` Season Number for the episode
- `Episode Number:` Episode Number for the episode

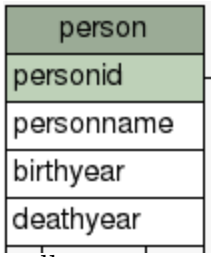


[8] `https://www.postgresql.org/docs/9.1/static/sql-copy.html`

We added checks for certain values to ensure to incorrectly formatted data is inserted into our tables. These include numerical checks for years, budget, gross box office, season and episode numbers.

6.2 `Person`

The `person` table is a sort of an base table for getting personal information relating to a person. Information available is:



- `PersonID:` A unique ID for each person
- `PersonName:` The person's name
- `BirthYear:` The person's birth year
- `DeathYear:` The person's death year

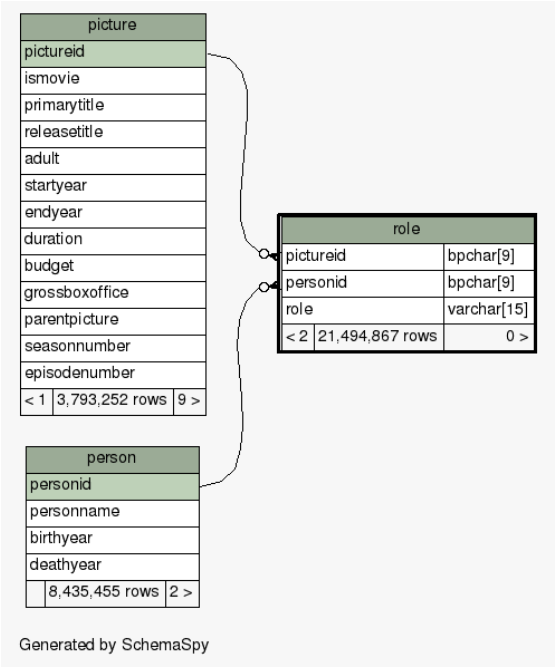
Again, we add `NOT NULL` checks for names, and numeric checks for years as well.

6.3 `Role`

This table gives the roles of people with movies as described earlier (hence we are not adding a redundant explanation here). We have set the default role to be Miscellaneous.

6.3.1 Relationship between `Role`, `Picture` and `Person`

We establish that `PictureID` and `PersonID` together compose as one foreign key, which results in a cascading deletion. Below is a tabular representation:



6.4 `Awards`

This table gives the awards that a person and/or a picture has won. We have specified the means of obtaining this data. The attributes in this table are:

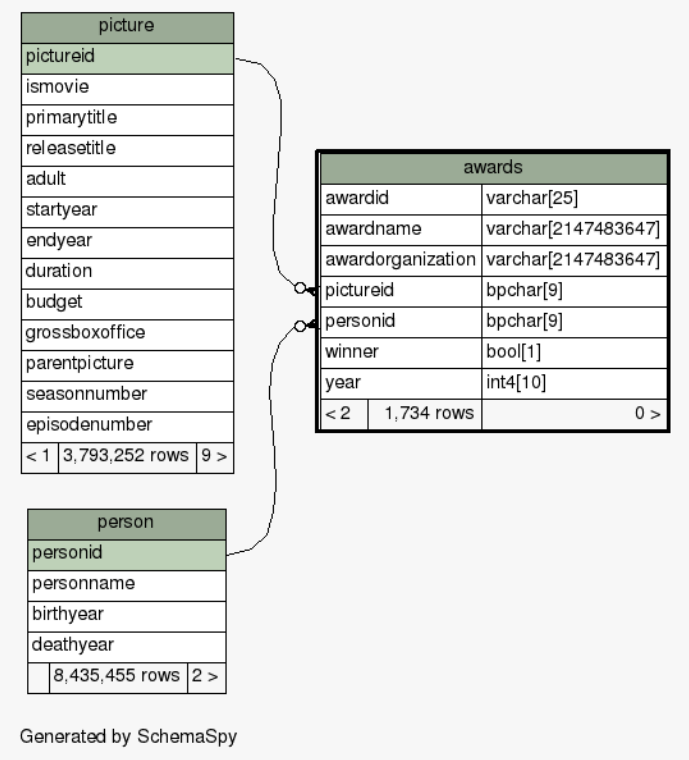
- `AwardID:` A unique ID per award
- `AwardName:` Name for the Award
- `AwardOrganization:` Organization giving the Award
- `PictureID:` Foreign key from `Picture` table
- `PersonID:` Foreign key from `Person` table
- `Winner:` A boolean specifying if the person/picture won the award or was only nominated.
- `Year:` Year of the Award

awards
awardid
awardname
awardorganization
pictureid
personid
winner
year

Checks are added in-place to ensure `NOT NULL` characteristics for select attributes and numeric check for the year.

6.4.1 Relationship between `Awards`, `Picture` and `Person`

Here there is no foreign key reference per se, since there can be multiple same persons for different awards. The awards table is basically a means to connect people, their contributions to television and theatre and awards received for their performances.



6.5 Minor Tables

These tables just contain more information about the movies. There are two tables for locations: one for filming location, which we took to be the production location after crawling this information. This is present in the table `filming_location`. Similarly, we have another table for release locations, which we obtained from the IMDB dataset (filename: `title.akas.tsv` in the dataset). This is named `release_location`. This have a link to `Picture` via `PictureID`.

We also designed tables for the languages that a movie has been released in: which is represented in `languages`. A similar table is also created for the genres of the movie, and is called `genres`.

Speaking of production, through crawling we identified the production companies associated with certain pictures. This information is stored in `production_company`. The company IDs for the production companies in the table are taken from the crawled data itself.

6.6 One final connection

The final representation can be viewed here. We have made use of SchemaSpy for helping generate this.