CSCI 3700/6070 Database Systems
CLASS PROJECT FALL 2019 (Total Points: 100)

Apps and Data - A study of IMDb dataset

Auburn University at Montgomery - Dept. of Comp. Sci., Montogomery, Alabama

Kelvin Gao

zgao1@aum.edu

Feb 12, 2020

Abstract

IMDb (Internet Movie Database) is an online database of information related to films, television programs, home videos, video games, and streaming content online – including cast, production crew and personal biographies, plot summaries, trivia, fan and critical reviews, and ratings. An additional fan feature, message boards, was abandoned in February 2017. Originally a fan-operated website, the database is owned and operated by IMDb.com, Inc., a subsidiary of Amazon.

As of May 2019, IMDb has approximately 6 million titles (including episodes) and 9.9 million personalities in its database,[2] as well as 83 million registered users.

IMDb began as a movie database on the Usenet group "rec.arts.movies" in 1990, and moved to the web in 1993 [1].

1 Objectives

In this project, students are supposed to learn:

1. How to import data to database management system (e.g. MySQL).
2. How to build an App (e.g. Desktop, Web, CMD apps, etc.) to interact with database;
3. Understand the Model-View-Controller (MVC) design method by
   - Create models (e.g. classes and objects) that can organize and represent the data in the database;
   - Create views to present data (e.g. web pages and texts).
   - Create controllers to submit transactions and get data from database, to full-fill models then views.

2 Method

2.1 Dataset - IMDb

Available from: https://datasets.imdbws.com/.

The description of the dataset refers [2].

2.1.1 IMDb Dataset Details

Each dataset is contained in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column. A '\N' is used to denote that a particular field is missing or null for that title/name. The available datasets are as follows:

title.akas.tsv.gz - Contains the following information for titles:

- titleId (string) - a tconst, an alphanumeric unique identifier of the title
- ordering (integer) – a number to uniquely identify rows for a given titleId
- title (string) – the localized title
- region (string) - the region for this version of the title
- language (string) - the language of the title
- types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning
- attributes (array) - Additional terms to describe this alternative title, not enumerated
- isOriginalTitle (boolean) – 0: not original title; 1: original title

title.basics.tsv.gz - Contains the following information for titles:

- tconst (string) - alphanumeric unique identifier of the title
- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc)
- primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release
- originalTitle (string) - original title, in the original language
- isAdult (boolean) - 0: non-adult title; 1: adult title
- startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year
- endYear (YYYY) – TV Series end year. '\N' for all other title types
- runtimeMinutes – primary runtime of the title, in minutes genres (string array) – includes up to three genres associated with the title

title.crew.tsv.gz – Contains the director and writer information for all the titles in IMDb. Fields include:

- tconst (string) - alphanumeric unique identifier of the title
- directors (array of nconsts) - director(s) of the given title
- writers (array of nconsts) – writer(s) of the given title

title.episode.tsv.gz – Contains the tv episode information. Fields include:

- tconst (string) - alphanumeric identifier of episode
- parentTconst (string) - alphanumeric identifier of the parent TV Series
- seasonNumber (integer) – season number the episode belongs to
- episodeNumber (integer) – episode number of the tconst in the TV series

title.principals.tsv.gz – Contains the principal cast/crew for titles

- tconst (string) - alphanumeric unique identifier of the title
- ordering (integer) – a number to uniquely identify rows for a given titleId
- nconst (string) - alphanumeric unique identifier of the name/person
- category (string) - the category of job that person was in
- job (string) - the specific job title if applicable, else '\N'
- characters (string) - the name of the character played if applicable, else '\N'

title.ratings.tsv.gz – Contains the IMDb rating and votes information for titles

- tconst (string) - alphanumeric unique identifier of the title
- averageRating – weighted average of all the individual user ratings
- numVotes - number of votes the title has received

name.basics.tsv.gz – Contains the following information for names:

- nconst (string) - alphanumeric unique identifier of the name/person
- primaryName (string)– name by which the person is most often credited
- birthYear – in YYYY format
- deathYear – in YYYY format if applicable, else '\N'
- primaryProfession (array of strings)– the top-3 professions of the person
- knownForTitles (array of tconsts) – titles the person is known for

2.1.2 Import data to database

You can use tools available online or simply write your own scripts to import the data into the database. However, your database schema should at least satisfies:

- **Data Type**. Use appropriate data types in the database. For example, 'string' type in the dataset can be converted to 'varchar' type in the database; years can be either stored in 'varchar' or 'date' type.
- **Null values**. The dataset contains null values that use '\N' to present. You should handle the null values during importing.
- **Keys and constrains.** Every table should have a primary key (either a single column or a set of columns). You should decide which column(s) to be used as the primary key. You also need to pay attention to the following foreign keys constrains:
  1. 'titleID', 'parentTconst' and 'tconst' in other tables should refer to 'tconst' in 'Title.Basic'.
  2. 'ncost' in other tables should refer to 'nconst' in 'Name.Basic'.
  3. You are free to add other foreign keys.
- **Array**. Some attributes/columns contain an array of values. You should consider how to store it in the database and how to handle it with your app. (Hint: you can use a separator like '#' to split the values and store them as a whole string; when using the controller of your app, just simply use token methods (e.g. JAVA: String.split("#")) to handle them.
- **Long String** Some attributes/columns contain a long string. You should consider how to store it in the database. One solution is to use text type instead of varchar/char, but it will affect the query speed significantly. The other solution is to create a separate table to store it but it may also lead to a overhead when joining two tables.

2.2 Test your data using SQL database (e.g. MySQL)

In this phase, simply execute some SQL statement to see if your data is successfully imported or not. For example, `SELECT * FROM Title.Basic;`.

2.3 Design SQL queries for the following questions.

1. Show the information of movies (Title, Region, Language, Year, Rating, Votes) which have average rating greater than 7 and number of votes greater than 1000.
2. Show the names of persons who have directed more than 10 movies.
3. Show the top 10 TV series (more than 1 season) which have most average ratings.
4. You need to design at least FOUR more queries which satisfy
   - Use aggregate functions, "group by" and "order by".
   - Use set operations.
   - Use nested/sub queries (either in SELECT clause, FROM clause or WHERE clause).
   - Use JOIN operations.
5. You may also receive bonus points (1 point each and up to 5 directly added to **your final grade**) if you can design queries using views and stored procedures.

2.4 Develop your App

Your app should follow the MVC style. For example (C++), the first query in Section 2.3, you may need to design a class for the model of movie:

```
class Movie {
private:
  string title;
  string region;
  string language;
  string year;
  double rating;
  int votes;
public:
  //...
}
```

Your view:

```
class MovieView {
priavate:
  Movie * result; //An array save the result from database
public:
  void printResult();

  void setResult(Movie * res);
}

void MovieView::printResult() {
  for(int i=0; i<N; i++) {
    printf("%s/t%s/t%s/t%s/t%.2f/t%d/n", result[i].title ...);
  }
}

void MovieView::setResult(Movie * res) {
  this->result = res;
}
```

And your controller:

```
class MovieController {
private:
    MovieView view;

public:
    bool getTopRatingMovies();
}

bool MovieController::getTopRatingMovies() {
    //Connect to Database;
    //Execute your SQL statement;
    //Get the result set ReSet
    Movie * list = new Movie[ReSet.count];
    for (int i=0;i<ReSet.cound;i++) {
        list[i].title = ReSet[i].get("title");
        ...
    }
    this->view.setResult(list); //Set the results to view
    this->view.printResult();   //And output them
}
```

2.5 Cross-Class project

This project is cross-linked to CSCI 4450/6400 Cloud Computing if

- You host your web app on AWS by using Lightsail or Elastic Beanstalk.
- You database is on the cloud, either EC2 or RDS.

This project is cross-linked to CSCI 4450/6400 Cloud Computing and CSCI 4500/6970 if

- You host your web service on AWS by using Lightsail or Elastic Beanstalk.
- You database is on the cloud, either EC2 or RDS.
- You build an mobile app (either iOS or Android) to get data from web service and show it on smartphone.

3 Deliverables

3.1 First deliverable

Due: Mar 28
Points: 30
Turn in a report with SQL statements and screenshots of the output results in Section 2.3. A template of report will be provided later.

3.2 Project Demo

On: Week 15 (Apr 23 and Apr 25)
Points: 40
Live demo of your app. Remember to have a backup plan for any exceptions.

3.3 Final report

Due: April 30 (Exam day)
Points: 30
This is a final report of your project. You should include your project management (task for each individual), your design of the app (MVC), etc. A template of report will be provided later.

References

[1] IMDb. Imdb datasets, available from "https://en.wikipedia.org/wiki/IMDb".

[2] Wikipedia. Imdb, available from "https://www.imdb.com/interfaces/".