CSCI620 Phase 2 : IMDB Dataset

Ketan Kokane
Rochester Institute of Technology
1 Lomb Memorial Dr,
Rochester, NY 14623
kk7471@rit.edu

Siddarth Sargunaraj
Rochester Institute of Technology
1 Lomb Memorial Dr,
Rochester, NY 14623
sxs2469@rit.edu

Ameya Nagnur
Rochester Institute of Technology
1 Lomb Memorial Dr,
Rochester, NY 14623
an4920@rit.edu

Kavya Kotian
Rochester Institute of Technology
1 Lomb Memorial Dr,
Rochester, NY 14623
kk2014@rit.edu

ABSTRACT

This *project* analyses the dataset used by IMDb. There is an ER diagram representing the entities constituting the dataset and relationships between them. Based on this ER diagram, we build a Relational Schema that helps us write queries to create tables for this database. We improve the database with indexing and create queries to provide efficient access of data to the users. Made use of python to translate tsv files to sql

Keywords

Data Management, ER Diagram, Relational Scheme, Functional Dependencies, Indexing

1. WHAT HAS BEEN DONE?

- Analysis of the entities in the IMDB dataset
- Convert the dataset entities to an ER diagram
- Improve upon the ER Diagram
- Create a Relational Schema
- Found out the functional dependencies in the dataset and to normalize the data
- SQL scripts to create and populate tables
- Test the schema performance with sql queries

2. ER DIAGRAM

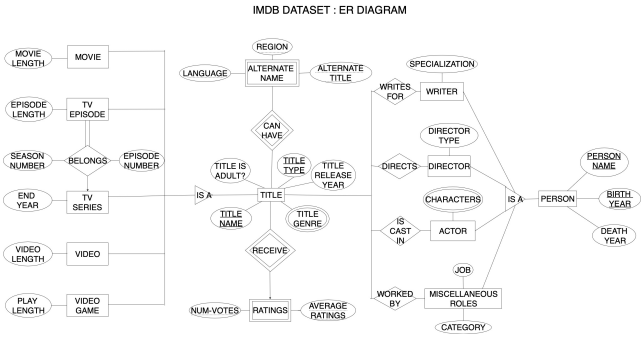

Figure 1: ER Diagram

3. ANALYSIS OF THE ENTITIES IN THE IMDB DATASET

The entities defined in the dataset are Title, Title Genres, Movie, Video, Video games, TV Series, TV Episode, Alternate Name, Ratings, Person, Writer, Director, Actor, Character, Miscellaneous Roles.

Every title has a Title name and Title type which are unique for every Movie, Video, Video games, TV Series and TV Episode. A title can belong to multiple Genres. A title can belong to multiple Genres and notifies if the content is adult. **IMDb provides data about Movie, Video, Video games, TV Series, TV Episode which are instances of the Title entity.**

Movie is a Title and gives the Movie Length. Video is a Title with an additional attribute Video Length. Video Games is a Title with an additional attribute Video Length. Every TV Episode belongs to a TV Series and a TV Series has multiple TV Episodes. Alternate Names and Ratings depend on Title and their attributes serve as Discriminators.**Imdb provides data about every cast members of every title**

Every Person has a Person name and Birth Year which are unique for every Person, Writer, Director, Actor and Miscellaneous Roles. A Person entity also tells us about the persons Death Year.

4. RELATIONAL SCHEMA

Title (Title_Name, Title_Type, Title_Release_Year, Title_Is_Adult?)
PK: (Title_Name, Tiltle_Release_Year, Title_Type)

Title_genres (Title_Name, Title_Type, Genre)
PK:(Title_Name, Title_Type, Genre)
FK: (Title_Name, Title_Type)

Movie(Title_Name, Title_Type, Movie_Length)
PK:(Title_Name, Title_Type)
FK:Title_Name, Title_Type)

Video(Title_Name, Title_Type, Video_Length)
PK:(Title_Name, Title_Type)
FK:(Title_Name, Title_Type)

Video_Games(Title_Name, Title_Type, Play_Length)
PK:(Title_Name, Title_Type, Play_Length)
FK: (Title_Name, Title_Type)

TV_Series (Title_Name, Title_Type, End_Year)
PK:(Title_Name, Title_Type)
FK: Title_Name, Title_Type)

TV_Episode (Title_Name, Title_Type, Episode_Length)
PK:(Title_Name, Title_Type)
FK: Title_Name, Title_Type)

Belongs(TV_Series_Name, TV_Series_Type, TV_Episode_Name, TV_Episode_Type, Season_Number, Episode_Number)
FK_TV_Episode = (TV_Series_Name, TV_Series_Type)
FK_TV_Series = (TV_Episode_Name, TV_Episode_Type)

Alternate_Name(Title_Name, Title_Type, Region, Language, Alternate_Title)
PK: (Title_Name, Title_Type, Region, Language, Alternate_Title)
FK: (Title_Name, Title_Type)

Ratings(Title_Name, Title_Type, Num-Votes, Average_Ratings)
PK: (Title_Name, Title_Name, Title_Type, Num-Votes, Average_Ratings)
FK: (Title_Name, Title_Name, Title_Type)

Person(Person_Name, Birth_Year,Death_Year)
PK: (Person_Name, Birth_Year)

Writer(Person_Name, Birth_Year, Specialization)
FK: (Person_Name, Birth_Year)

Writes_For(Title_Name, Title_Type,Person_Name, Birth_Year)
FK_Person: (Person_Name, Birth_Year)
FK_title: Title_Name, Tiltle_Release_Year, Title_Type)

Director(Person_Name, Birth_Year, Director_type)
FK: (Person_Name, Birth_Year)

Directs(Title_Name, Title_Type,Person_Name, Birth_Year)
FK_Person: (Person_Name, Birth_Year)
FK_title: Title_Name, Title_Type)

Actor(Person_Name, Birth_Year)
FK: (Person_Name, Birth_Year)

Character(Person_Name, Birth_Year,Character)
FK: (Person_Name, Birth_Year)

Is_Cast_In(Title_Name, Title_Type,Person_Name, Birth_Year)
FK_Person: (Person_Name, Birth_Year)
FK_title: Title_Name, Title_Type)

Miscellaneous_Roles(Person_Name, Birth_Year,Category, Job)
FK: (Person_Name, Birth_Year)

Worked_By(Title_Name, Title_Type,Person_Name, Birth_Year)
FK_Person: (Person_Name, Birth_Year)
FK_title: (Title_Name, Title_Type)

5. FUNCTIONAL DEPENDANCIES

5.1 Assumptions

- (Title_Name, Title_Type) be represented as A
- (Title_Name, Title_Type) for TV_Series be represented as A_{series}
- (Title_Name, Title_Type) for TV_Episodes be represented as$A_{episodes}$
- (Person_Name, Birth_Year) be represented as B

5.2 Dependencies

A→ Title_Release_Year, Title_Is_Adult?
A → Movie_Length
A → Video_Length
A → Play_Length
A → End_Year
A → Episode_Length
$A_{series} \rightarrow A_{episodes}$
$A_{episodes}$ → Season_Number, Episode_Number
B → Death_Year
B → Specialization
B → Director_type
B → Character

B → Category,Job

6. DATABASE POPULATION

Attached *create_all.sql* and *populate_all.sql* file which are generated using the python script which parses the entire data-set of IMDB and samples 10000 titles from and loads the dataset.

7. PYTHON SCRIPT TO READ AND LOAD THE DATASET

Attached *code.py* file which uses pandas framework to analyze the dataset and based on the functional dependencies generates the sql files.

8. QUERIES

8.1

8.1.1 Description

The query creates a join of the tables movie and Titlegenres to get the movie titles in the genre 'Comedy'.

8.1.2 Query

```
SELECT M.titlename,
       G.genre
FROM   movie M,
       titlegenres G
WHERE  M.titlename = G.titlename
       AND G.genre LIKE 'Comedy';
```

8.1.3 Output

TitleName	Genre
41 el hombre perfecto	Comedy
A Bizarre Love Triangle	Comedy
Adventures of Death	Comedy
All Wrong	Comedy
America's Most Retarded	Comedy

(few output values)

8.1.4 Query justification

Joins 2 tables based on titlename i.e. an attribute with most unique entries in the database and fetches information based on genres.
$Duration/FetchTime$: $0.359sec/0.000054sec$

8.2

8.2.1 Description

The query creates a join of the tables belongs (Maps TvSeries name as S_TitleName with it's episode names as E_TitleName), Titleratings and tvepisode to get the total number of votes for tv series based on the ratings for it's episodes.

8.2.2 Query

```
SELECT B.s_titlename,
       Sum(R.numvotes)
FROM   belongs B,
       titleratings R,
       tvepisode E
WHERE  B.e_titlename = E.titlename
       AND E.titlename = R.titlename
GROUP  BY B.s_titlename;
```

8.2.3 Output

S_TitleName	sum(R.NumVotes)
Ferris Bueller	8
Neighbours	8

8.2.4 Query justification

Joins 3 tables again based on titlename but only of episodes and tv series. We use the group by clause to get episodes for every tv series and sum their votes making it more complex than simply fetching votes or ratings for a title.
$Duration/FetchTime$: $0.041sec/0.000011sec$

8.3

8.3.1 Description

The query creates a join of the tables titleAlternateName and Title to get the alternate titles for any Movie, TvSeries or other type of titles for a particular region. ("India" here)

8.3.2 Query

```
SELECT  T.TitleName,A.alternate_title
FROM   titlealternatename A,
       title T
WHERE  T.titlename = A.titlename
       AND A.region = 'India';
```

8.3.3 Output

TitleName	alternate_title
There & Back	Kalia

8.3.4 Query justification

Joins 2 tables and uses a table that provides region specific information. The join is done again on titlename for any type of title.
$Duration/FetchTime$: $0.0034sec/0.0000088sec$

8.4

8.4.1 Description

- The query is a nested query with the inner query joining the tables Titleratings, director and title_director (Maps director names with title names they have worked for) to fetch the average ratings for all titles for which director named Aaron Kodz has worked.
- The outer query fetches all the directors and the titles they have worked for if the title has an average rating > than all ratings fetched by inner query.

8.4.2 Query

```
SELECT R.titlename,
       D.personname,
       R.averageratings
FROM   titleratings R,
       director D,
       title_director TD
```

```
WHERE  TD.titlename = R.titlename
       AND D.personname = TD.personname
       AND R.averageratings
         >ALL (SELECT TR.averageratings
               FROM   titleratings TR,
                      director DR,
                      title_director TDR
               WHERE  TR.titlename = TDR.titlename
                      AND DR.personname = TDR.personname
                      AND DR.personname = 'Aaron Kodz');
```

8.4.3 Query justification

Has a nested query. Inner query joins 3 tables based on 2 different attributes that fetches average rating based on titles worked on by a specific director. Outer query joins the same three tables and checks for a value larger than all fetched by inner query. Equivalent to a nested loop in any programming language.
$Duration/FetchTime$: $0.214sec/0.286sec$

8.4.4 Output

TitleName	PersonName	AverageRatings
111 Girls	Bijan Zmanpira	74
111 Girls	Nahid Ghobadi	74
41 el hombre perfecto	Pepe Romay	12
A Aa E Ee	Sabapathy	26
A Bizarre Love Triangle	Mu-yeong Lee	131

(few output values)

8.5 Query justification summary

We have tested queries with complexities of joining 2 or more tables, with a nested structure and with group by. We have covered all key tables hence covering a major chunk of the data ranging from a simple fetch from a 2 table join to a comparison with all entries of a table returned by an inner query. The remaining tables like Writer can easily be switched with Director like in query 4 but the result would be similar. Hence, all types of data is covered and less likely but complex queries are tested along with simple ones.

9. INDEXES

Used Cluster Index generated by the MySQL as the only indexes for the database, as design of the database allwoed to query the majority of the data just by using cluter index.

10. WORK DISTRIBUTION

Task	Handled By
Analysis of the entities in the IMDB dataset	Ameya, Kavya
Convert the dataset entities to an ER diagram	Ketan, Siddarth
Improve upon the ER Diagram	Ketan, Siddarth
Create a Relational Schema	Ameya, Kavya
Found out the functional dependencies in the dataset and to normalize the data	Ameya, Kavya
SQL scripts to create and populate tables	All
Test the schema performance with sql queries	All

11. CONCLUSION

Analyzed the imdb dataset and learned how to model the entities in the dataset, relate two entities using relation, mapping the cardinalities of the relations.

The IMdb dataset contains data about 12 different categories of titles, which can be relased in multiple regions in different languages. Maps persons who works in the movie.