Explore Databases Application Final Report
INF551
Mingjun Liu(1321657359)
Minghong Xu(3989201972)

Overview

Our final web application has the website: https://exploredatabase.herokuapp.com/

Project demo Youtube link: https://www.youtube.com/watch?v=iTNC5oMu_8E

More related materials included in the google drive. Link of google drive: https://drive.google.com/open?id=1bkkWJfZOVSNMaxs8ElMGyJMRAVJ5MYBS

Project idea

In this project, we are going to implement a web application that allows our user to explore the databases we choose. The user is able to search one or multiple keywords within any one of our databases and navigate between the tables by hyperlinks. The application will involve techniques about both front end and back end. We need to design an user interface that allows the user to choose the database to explore and submit the keywords to the back end, which employs forms from html. The hyperlink of the foreign key in the tables should allow the user to jump to the tables with the foreign key as an primary key, which means each table should have at least one foreign key relationship to the other tables. Back end responsibility includes importing databases from MySQL and uploading databases to Firebase, realizing the import function, search function and the navigation function, which are described with detailed explanation in the following parts. Since both of the members have lacked experience on designing user interfaces, we used the web application helper herokuapp.com to complete the UI. At this time, the three databases we chose are World, Movies, and FIFA World Cup.

The first database is the world database provided by the professor. The database contains three tables: city, country and countrylanguage. Code is the primary key in the country table, which is named as CountryCode and acts as foreign key in city table and countrylanguage table.

The second database was adapted from MovieLens, which is a movie recommendation service. The database contains three tables, which are movies, tags and ratings. The movie table includes movieId, movie titles and genres the movies belong to. The tags table includes userId, movieId and tags given to the movie from the users. The rating table includes userId, movieId and rating to the movies. User ids are consistent between ratings table and tags table, and movie ids are consistent among all three tables.

movies

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Chil
2	Jumanji (1995)	Adventure|Children|Fanta
3	Grumpier Old Men	Comedy|Romance
4	Waiting to Exhale (	Comedy|Drama|Romance
5	Father of the Bride	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (19	Adventure|Children
9	Sudden Death (19	Action
10	GoldenEye (1995)	Action|Adventure|Thriller
11	American Presider	Comedy|Drama|Romance
12	Dracula: Dead and	Comedy|Horror
13	Balto (1995)	Adventure|Animation|Chil
14	Nixon (1995)	Drama
15	Cutthroat Island (1	Action|Adventure|Roman

tags

userId	movieId	tag	timestamp
2	60756	funny	1445714994
2	60756	Highly quotable	1445714996
2	60756	will ferrell	1445714992
2	89774	Boxing story	1445715207
2	89774	MMA	1445715200
2	89774	Tom Hardy	1445715205
2	106782	drugs	1445715054
2	106782	Leonardo DiCapı	1445715051
2	106782	Martin Scorsese	1445715056
7	48516	way too long	1169687325
18	431	Al Pacino	1462138765
18	431	gangster	1462138749
18	431	mafia	1462138755
18	1221	Al Pacino	1461699306

ratings

userId	movieId	rating	timestamp
1	1	4	964982703
1	3	4	964981247
1	6	4	964982224
1	47	5	964983815
1	50	5	964982931
1	70	3	964982400
1	101	5	964980868
1	110	4	964982176
1	151	5	964984041
1	157	5	964984100
1	163	5	964983650
1	216	5	964981208
1	223	3	964980985
1	231	5	964981179

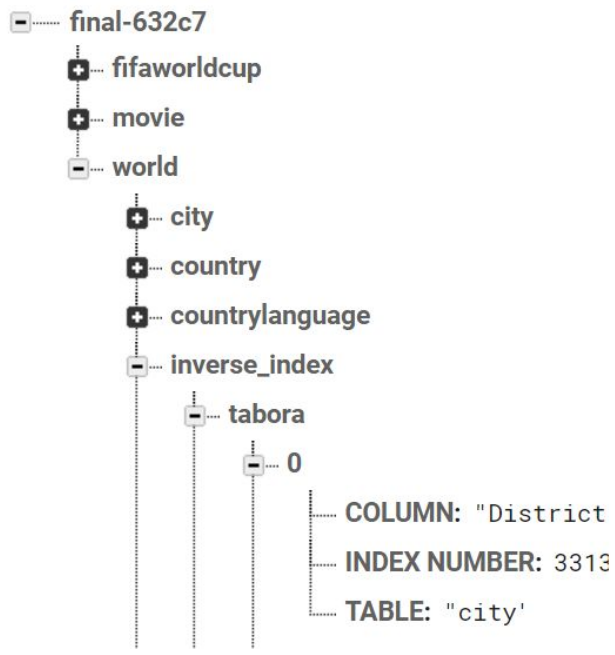
The third database is the FIFA World Cup database from Kaggle Data. The database contains three tables: WorldCupMatches, WorldCupPlayers and WorldCups. This database has a more complicated foreign key relationship between the tables. For example, Year is the primary key in the WorldCup table, and the foreign key in WorldCupMatches table; matchID is the primary key in the WorldCupMatches table, and the foreign key in WorldCupPlayers.

Working components

Part 1: import.py

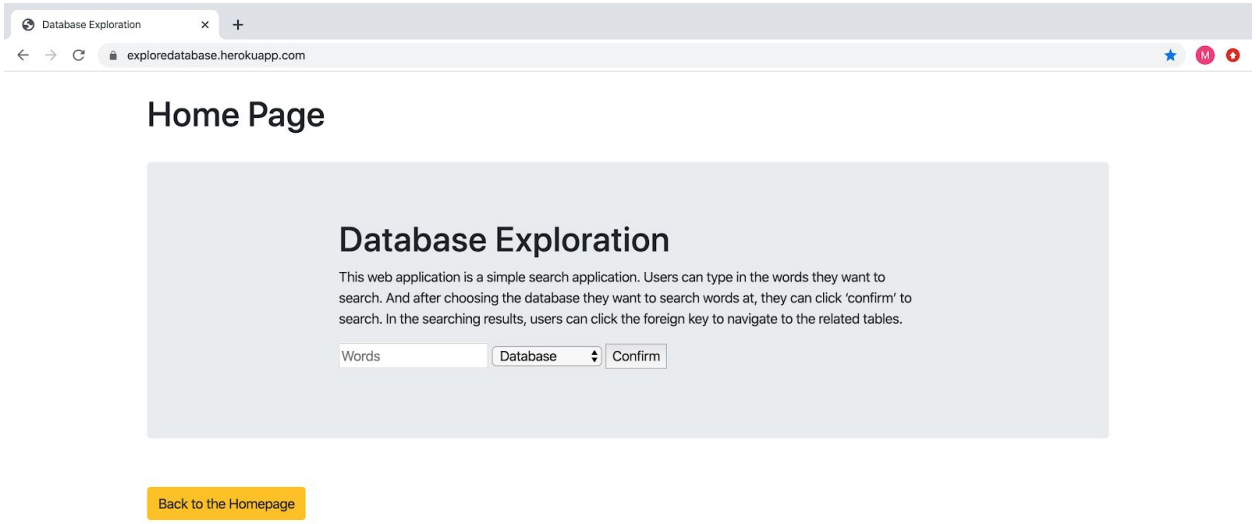
After running the execution syntax, the program will show “inverse index uploaded to Firebase” if everything works well. The link to Firebase will create a node “world” that contains all the tables in the database and an inverse_index that contains the index location of a word.

```
C:\Users\marga\Dropbox\USC\2020SPRING\INF551\project>python import.py world world
use information_schema
select TABLE_NAME from tables where TABLE_SCHEMA = 'world'
select COLUMN_NAME from COLUMNS where TABLE_NAME = 'city' and TABLE_SCHEMA = 'world'
select COLUMN_NAME from COLUMNS where TABLE_NAME = 'country' and TABLE_SCHEMA = 'world'
select COLUMN_NAME from COLUMNS where TABLE_NAME = 'countrylanguage' and TABLE_SCHEMA = 'world'
use world
select * from city
select * from country
select * from countrylanguage
****Return database data****
***upload table city ***
***upload table country ***
***upload table countrylanguage ***
***Inverse index uploaded to Firebase***
```

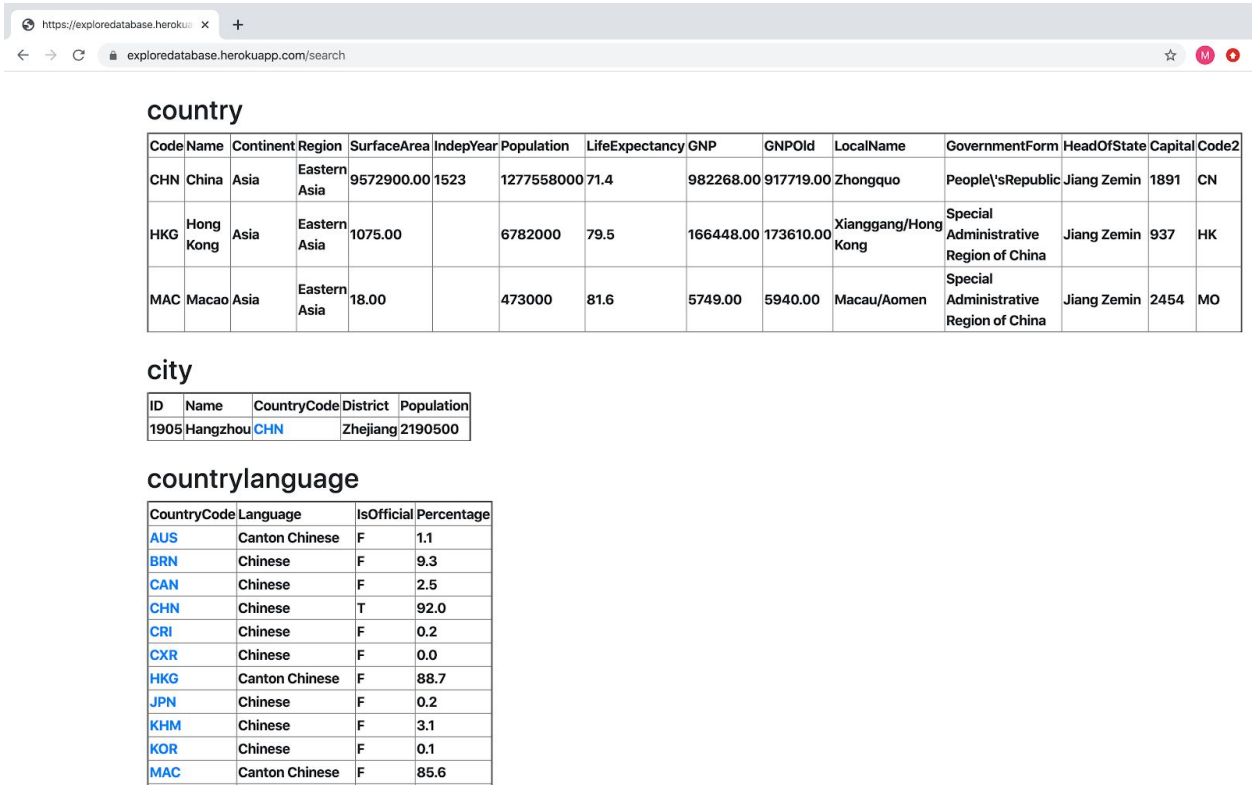


Part 2: Web Application

Our main page:

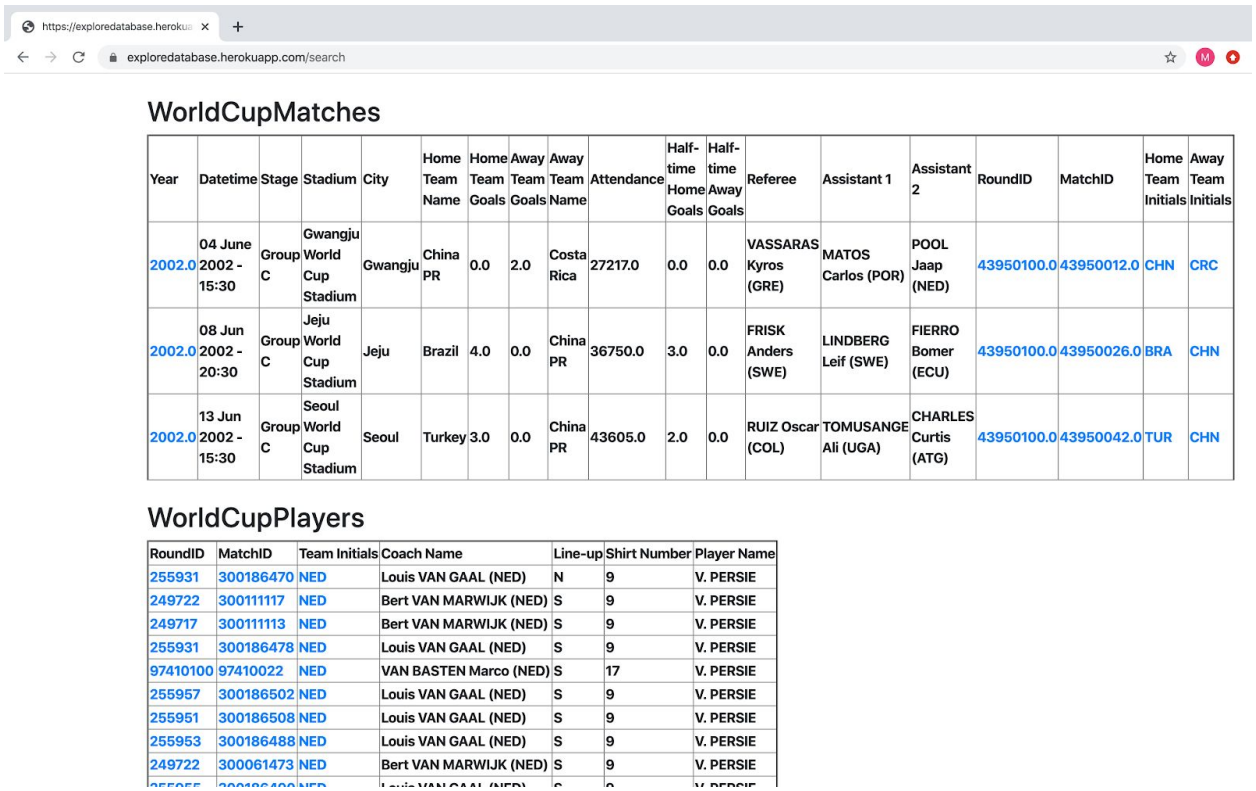


Search result for 'China Chinese hangzhou' of World database:



Search result for ‘iconic fiction’ of Movie database:

Search result for ‘China persie’ of FIFA World Cup database:



Implementation details

Most of our work is built based on python. For the data, we write an import file, which uses python to script the data from MySQL, clean the data and upload the data to Firebase, so that the web application can retrieve data from Firebase directly without downloading data to the local disk. While reading the databases into python, since we don't know the columns of each table, we need to use the information_schema database to help us read the columns. We created a list that contains the name of the tables and corresponding names of the columns. After that, the cells of the database are saved by lists in memory. To make locating the words easier, when creating the index dictionary for each database, we used the index number as the primary key, which was generated by Firebase when we uploaded the database.

Furthermore, we write backend functions with python including searching words and retrieving data from tables with specific keys. In order to make the search function work, we also create an index table at Firebase for each database, which creates an index for each word appearing in a database. Some tuples of the table may contain more than one search word the users type in. So we also rank the tuple returned by the table with the times of search key appeared.

The last step is to realize the application on the website. We use the Flask Framework in python to write the website. Flask is a very useful package in python. It transfers many complex HTML functions and connection stuff to python code. Although Flask helps us a lot, we still write many HTML files to implement the templates of web pages. We also create hyperlink for every word functioned as foreign key that users can click to navigate to other tables with the key. And users can navigate continuously when there are foreign keys in the returned table. For the words that are foreign keys, they show with blue color. Finally, we use heroku to publish our website.

Performance analysis

We use the time function in python to record the time each search used. And use the time to analyze the query processing performance. As for the query mentioned above, we repeated several times to increase the reliability.

Search for 'China Chinese hangzhou' of World database takes 6.49s ~ 6.71s
Search for 'iconic fiction' of Movie database takes 0.93s ~ 1.27s

Search for ‘China persie’ of FIFA World Cup database takes 0.95s ~ 1.03s

It is obvious that searching in the world database takes much more time. The reason is that we have 3 searching words for that database. We tried to search only two words, and the time decreased to around 1s. Therefore, in general, all three databases behave similarly, which has nothing to do with the sizes of databases. The reason behind that is we use python to get data from Firebase, and transfer to the data type of dictionary. Dictionary is a hash table. Therefore, the time to get data is linear.

To improve the performance, we might find enhancement at the number of going through the database. At our existing searching code, for each search word the user types in, the function would go through the whole database one time. So if the user searches 10 words, the function would go through the database 10 times, which is a huge consumption. One way to improve is to cache all the search words in memory, and search them together in one iteration, that would save a lot of time.

Responsibility and work

There are two members in our team: Mingjun Liu and Minghong Xu. We do not have any abilities related to web development or mobile apps before. So we learnt web development by ourselves. We design and develop the web application together.

Conclusion

The database exploration web application consists of the knowledge we learned during the semester, and let us apply different kinds of data management techniques together. During the coursework, we learned how to download the database from MySQL into a csv file and upload a csv file and create an index for the keywords into Firebase. This project combines these skills together to realize the data management in this application. It also allows us to learn how HTML works. Although both of us have no experience working on user interfaces, we design our search and navigation program with the help of herokuapp. We are now able to navigate the tables by the foreign key relationships, and also improve the performance of the search program by creating a reference table to keywords. The experience of developing the web application let us understand how each data management technique works and how to apply them on the databases based on our needs.

Source:

https://www.kaggle.com/abecklas/fifa-world-cup
http://files.grouplens.org/datasets/movielens/ml-latest-small-README.html
https://dev.mysql.com/doc/index-other.html