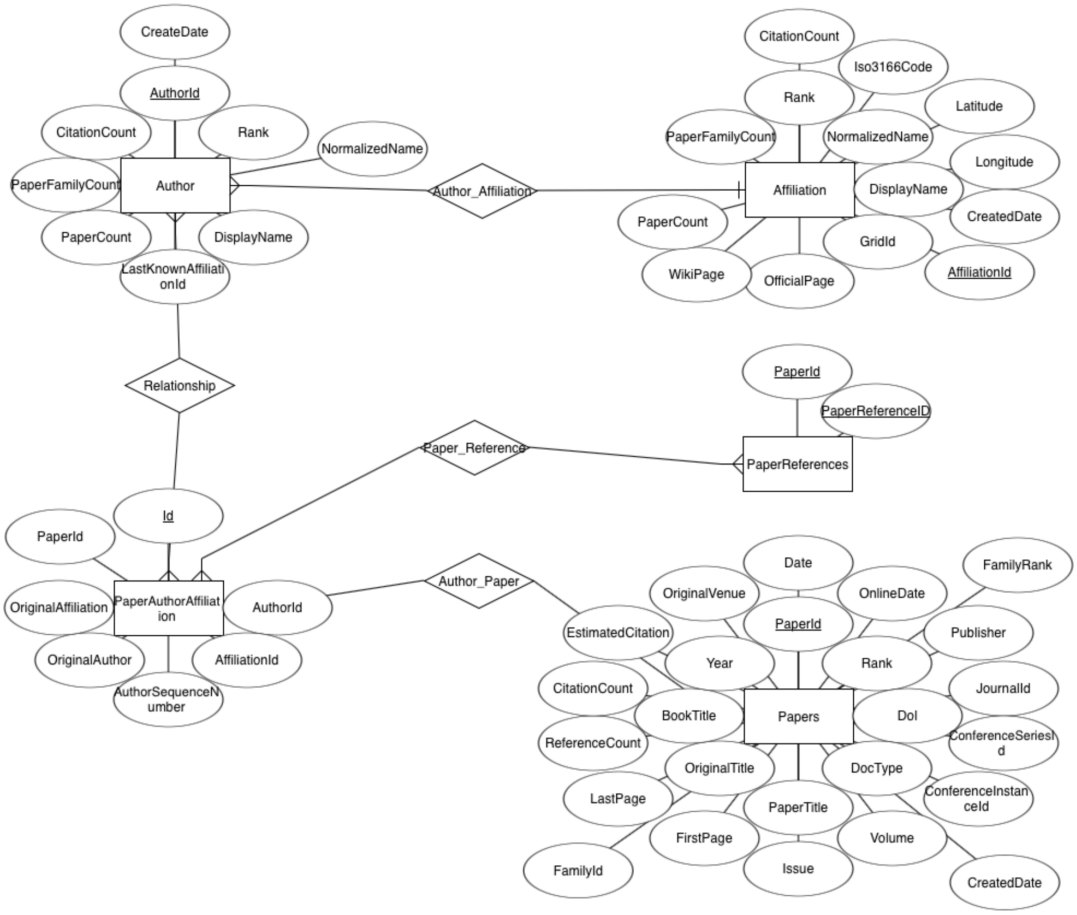
Flask-React-App Report

Jingtang Ma
jm58@illinois.edu
1/14/2022

MySQL(database)

ER Diagram



Database Design

I just referred the part of the design of the Microsoft's database but I designed the one which good fit for this task, the ER diagram is above. The idea is that I used LastKnownAffiliation as a foreign key of Author's table to Affiliation's table. I assumed each author is associated with exactly one affiliation. The PaperAuthorAffiliations' table links authors to papers, I used it to connect Author to Paper.

Given five tables such as: Author, Paper, Affiliations, PaperAuthorAffiliations, and PaperReferences. However, for task 1 and task 2, I did not use all of DB which is given:

For task 1, I used two tables: PaperAuthorAffiliations and PaperReferences. The relationship between there is Many–to–Many because a paper can be cited by multiple papers and a reference can be used by multiple papers as well. I could find all papers written by one author through PaperAuthorAffiliations, I could find all references cited by one paper through PaperReferences.

For task 2, I used two tables: Author and PaperAuthorAffiliations. The relationship between these is One–to–Many because each author is associated with exactly one affiliation, an affiliation can have multiple authors such as [a1 a2 a4…] but an author is only associated with one affiliation. I could find all papers written by one author through PaperAuthorAffiliations, I could find all co–authors from one paper through PaperAuthorAffiliations, I could find all author associated with one affiliation through Author.

Database Implementation

First, I used MySQL as my database, it created by "create_db.py". This python file can only be run one time to create a database, I will not use it anymore for later process.

Second, I used SQLAlchemy to create seven tables which are Author, Paper, Affiliations, PaperAuthorAffiliations, PaperReferences, UserInput, and UserInputt. The first five is given by professors, the UserInput stored users' inputs from task 1, the UserInputt stored users' inputs from task 2.

In the end, I imported all necessary data of Author, Paper, Affiliations, PaperAuthorAffiliations, and PaperReferences from the given .txt files, all SQL is in "add_db.py". This file can only be run one time to import data to the database, I will not use it anymore for laster process.

Flask(backend)

Task 1

Algorithms

1. Searched and saved all data which author_id equals the one that a user's input from PaperAuthorAffiliations table.
2. Searched and saved all paper_ids which are written by the author to a list.
3. Used the paper_id list, saved all references from PaperReferences table.
4. Searched and saved all the references to a dictionary to count the frequency
5. Sorted the dictionary by the value from big number to small number
6. Output Top 5 references

Logic

There are three function for this task in the backend: get_top_cited_papers, todo_serializer, and submit. get_top_cited_papers is the main function to compute what we need, the submit function can get users' inputs and post the inputs to the UserInput database. todo_serializer function does serializer for the results.

Task 2

Algorithms

1. Searched and saved all authors associated with the input affiliation to a list.
2. Searched and saved all papers written by these authors to a list.
3. Searched and saved all co-authors through the list of the papers.
4. Searched and saved all affiliations which contain these co-authors to a list.
5. Saved these affiliations to a dictionary and count the frequency.
6. Sorted the dictionary by the value from big number to small number.
7. Output several top institutions.

Logic

There are three function for this task in the backend: get_top_affiliation
, todo_serializer2, and submitt. get_top_affiliation is the main function to compute what we need, the submitt function can get users' inputs and post the inputs to the UserInputt database. todo_serializer2 function does serializer for the results.

React(frontend)

Configurations

In package.json, set "proxy" to "http://127.0.0.1:5000/" which is local address and connected to backend(Flask).

Card.js

It shows the results to user.

Form.js

It gives users an indication to input an author's id or an institution's id. When users finished, they can click "submit" button to submit what they inputted.

TodoPage.js

There are two TodoPages, the one is for task 1 and the another is fir task 2.

App.js

It contains two TodoPages and packages then to the frontend.

How to Run the App

1. Install MySQL, Python, Flask, Node.js, and React-app.
2. Configure virtual environment.
3. Build Flask APP
4. Create database by running “create_db.py”
5. Create database table by running db.create_all()
6. Import data by running “add_db.py”
7. Run Flask
8. Run React
9. Have fun!!!

Where can be Improved

Database

I did not use all of the data from the database, so my app only can search by IDs and get IDs for result. If a feature users want to use more data, you can implement them by using more data from the database such as paper names, author names, and affiliation names.

React(frontend)

According to the requirement, beauty is not mandatory. So I did not spend much time on the frontend. My app can do all the required tasks, but it does not good on useable and beauty. If a feature users want a pretty app, it can be implement through React.

Clarification

I did not import all data from the .txt files because it takes long time to import all data(.txt) to MySQL because the performance of my laptop is not good. I sent the email to Ashutosh and I got approve.

I did not allow my app to search all data for task 2 because it takes long time as well. In stead, I loop 10 times to get results for demo.

Due to time reason, I did not make this app what I expected it to be, I just finished all of the requirements you listed. If you want me to make this app perfect, please let me know and I will do it later.

References

https://urldefense.com/v3/__https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/__;!!DZ3fjg!r5AQ5xrTkDYeZyrlTCKHB1xeo5P4mKi27uhpM29UORc1qQploBRyEyuw4UbjuongNg$

https://urldefense.com/v3/__https://docs.microsoft.com/en-us/academic-services/graph/reference-data-schema__;!!DZ3fjg!r5AQ5xrTkDYeZyrlTCKHB1xeo5P4mKi27uhpM29UORc1qQploBRyEyuw4UbRcolcuA$

https://urldefense.com/v3/__https://drive.google.com/file/d/1xZyYrChZ_SwU5lA4YokdyL5kvCTScExP/view?usp=sharing__;!!DZ3fjg!r5AQ5xrTkDYeZyrlTCKHB1xeo5P4mKi27uhpM29UORc1qQploBRyEyuw4UYu6eD1QQ$