Project Progress Report

‘EmojiQL’

Name: **Aaryan Akshay Shah**
USC ID: **5532714539**
Date: **10/12/2023**

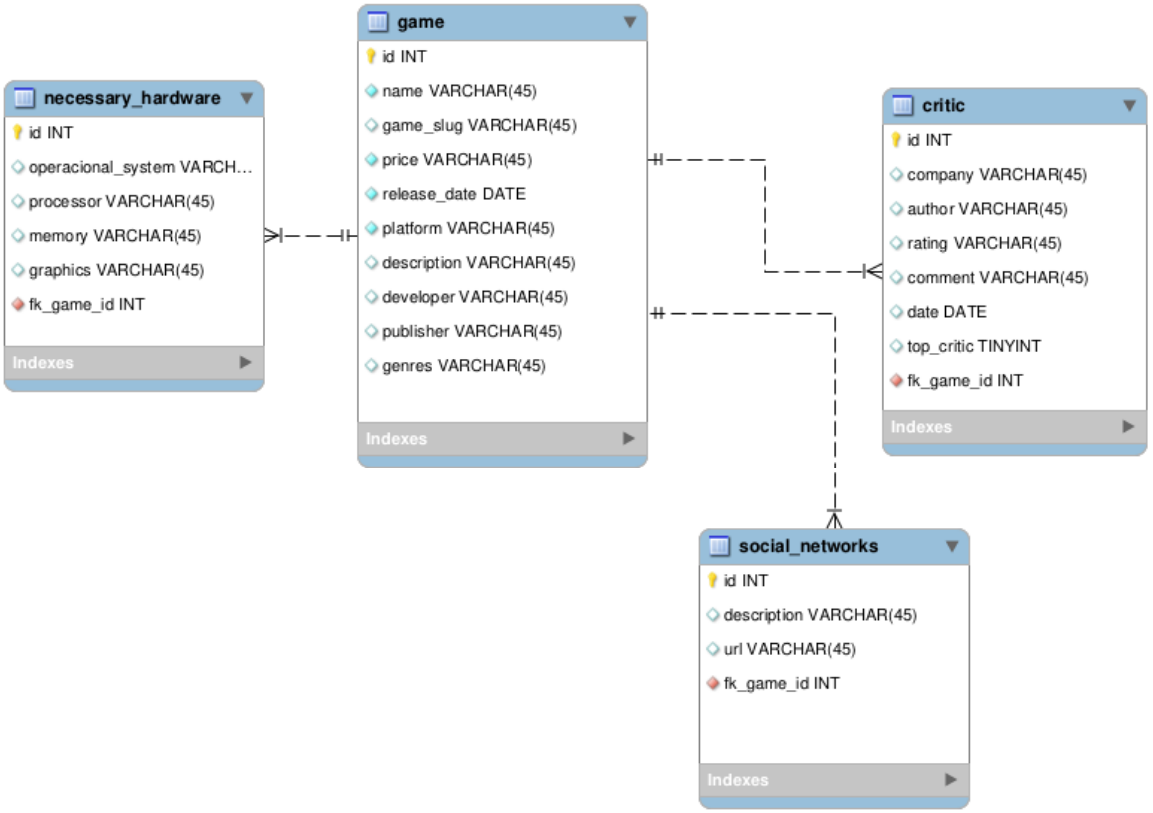
Data Cleaning:

Dataset link - 🎮 Epic Games Store Dataset (kaggle.com)

Earlier when I selected a dataset it had 6 tables. Following is the ER diagram for the original dataset.

Later after analyzing the data, I realized that there is no need to keep tables ‘tweets’ and ‘twitter_accounts’ since there is no direct connection between these two tables and other tables except ‘games’. Tables ‘necessary_hardware’, ‘critic’, and ‘social_networks’ are not related with other tables which I am going to remove.

Following is the ER diagram after removing the tables:



I have also checked for the null values using python and removed all them. I also got the insights about the dataset (eg. what is the shape of the dataset, etc.)
P.S. This is purely done just to get the data about the data(basically metadata). This is not used anywhere in the working of the project.

Defining the Emojis for EmojiQL :)

I am using the python **ast library** (Abstract Syntax Trees). The AST library in Python is a module that allows you to parse Python code into an abstract syntax tree (AST). An AST is a tree-like representation of the code's structure. It can be used to analyze the code, generate new code, or perform other tasks. The AST library provides a number of functions for parsing Python code into an AST. The most important function is the ast.parse() function. This function takes a string of Python code as input and returns an AST object that represents the code's structure.

Following is the code which I am using

Python ast library:

```
emojiql_grammar = {
    ' 🔍 ': 'SELECT',
    ' 📂': 'FROM',
    ' 🎯': 'WHERE',
    '📊 ': 'TABLE',
    '📎': 'AND',
    ' 🔸 ': 'OR',
    '🌟': 'LIKE',
    '🗑️': 'DELETE',
    '🔑': 'PRIMARY KEY',
    '➡🔑': 'FOREIGN KEY',
    '🤝': 'INNER JOIN',
    '👈🤝': 'LEFT JOIN',
    '👉🤝': 'RIGHT JOIN',
    '👈🤝👉': 'OUTER JOIN',
}
```

Above is just a gist of how the dictionary of emojis will look like and will be mapped using the ast library. I have figured out what other emojis need to be mapped to what functionality but haven't implemented yet just to begin with basics and get the understanding of overall working.

How data will be stored…

I am storing the data as CSV (comma separated values) and using the in-built ‘csv’ python library. To offload the datastore work to my own code, I will implement the fundamental functions of table creation insertion, deletion, update using file pointer operations. There will be an issue of atomicity and multiple concurrent operations sharing the pointing to the same file pointer, but it is a problem that is realistically not in our scope of study.

One more thing is to track multiple tables(in our case CSV files) in a metatable which will help us in queries which require us to access multiple tables, such as a join query. A more quicker way would be just to have a hashmap initialized with the file locations, so we don't need to worry about the physical location of the table rather than just proceed with the operations on the table.

To further optimize my datastore, I intend to index them. I hope to use B+ trees to index each table to improve the time complexity of the operations of updation and deletion.

Challenges faced till now…

- **Making a parser**
  - If we are introducing any of the keywords that have two separate tokens but are considered as a single node for the syntax tree, we are forced to check the next token for every token thus increasing parsing time. This took me a while to figure out what needs to be done. I hope that what I have implemented works in the overall setting. And ofcourse, if I am stuck at some point I will definitely contact the TAs to keep moving forward in the project.

- **Defining particular emojis for a particular query**
  - Initially it was difficult to decide what emoji needs to be mapped with what token because I had to select such emojis which were easy to remember and would directly correlate with the actual SQL query. It seems that I have made a conclusion on the list of emojis I need to use for this project. I have also mentioned above a few emojis which I am using.

Conclusion

I am confident that I will be able to complete the EmojiQL project by the end of the semester. I am excited to continue working on the project and to add new features to EmojiQL. I believe that EmojiQL has the potential to be a valuable tool for users who want to query data without having to learn traditional SQL.