Midterm Progress Report

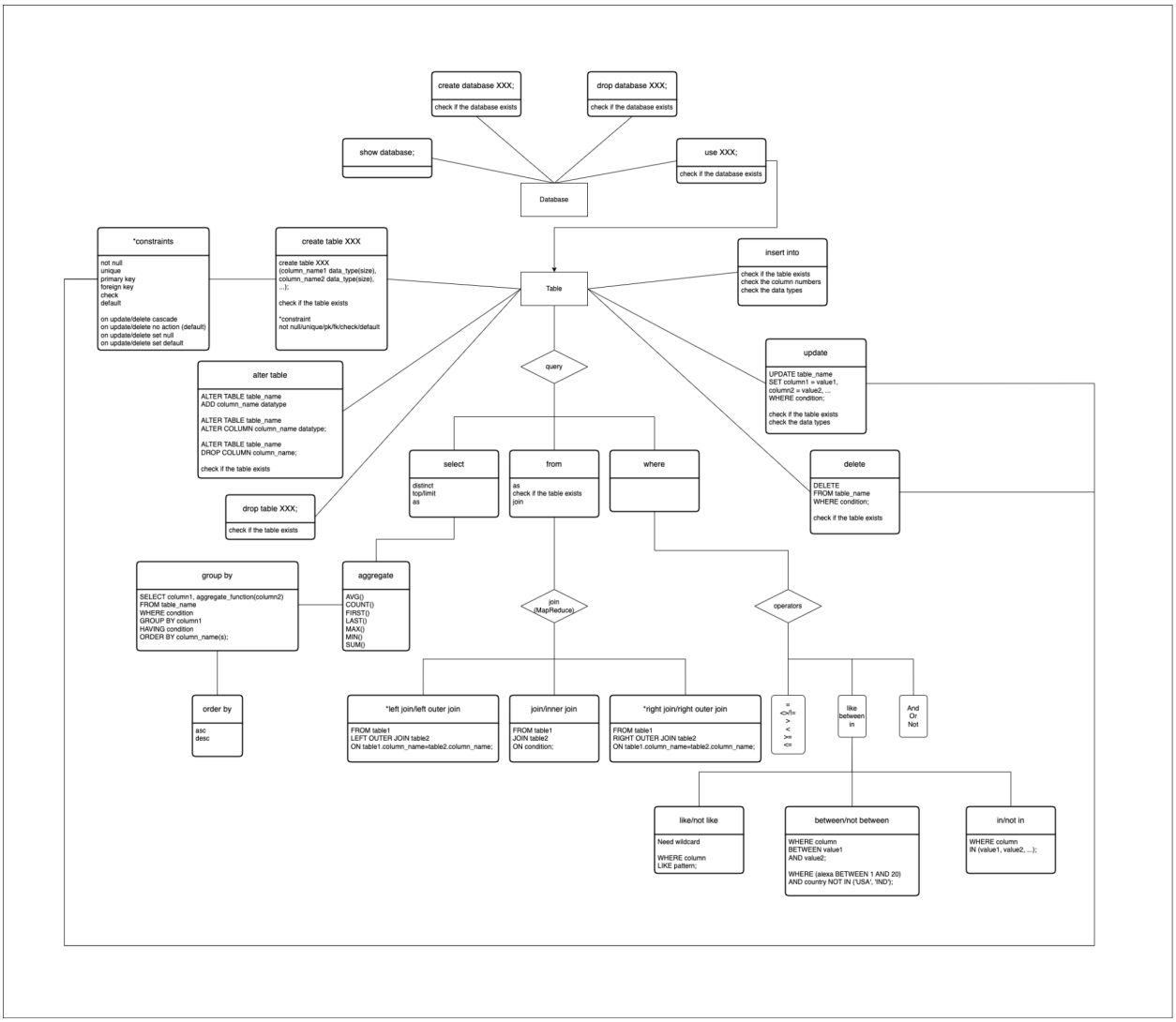
Group Members: Xuesong Shen (xuesongs@usc.edu), Haoze Zhu (haozezhu@usc.edu)
Class Session: Tuesday Afternoon

Progress:

1. Functions

We use a relational database as an example to demonstrate the functions to be implemented as the picture below. The query language (MySQL) in this diagram is only used for demonstration. Our query language will be different from this.

There are two levels, database level, and table level, connected by the command “use”. We will implement CRUD functions, as well as projection, filtering, join, grouping, aggregation, and ordering functions as required.



2. System Structure

We also finished the design of our system's structure.

ourDB(Folder):
——Code for SQL (Folder)
——Code for NoSQL (Folder)
——SQL(Folder)
    Database A(Folder)::
        Table1.csv
        Table2.csv
        SQL_running(Folder):
            Table1_running.csv
            Table2_running.csv
——NoSQL(Folder)
    Database B(Folder):
        Collection3.json
        Collection4.json
        NoSQL_running(Folder):
            Collection3_running.json
            Collection4_running.json

3. Datasets

For the two database systems, we found datasets from Kaggle for the relational database and non-relational database respectively.

3.1 Relational Dataset

NBA Games Data
This dataset was collected to work on NBA games data, created using the nba stats website. There are 5 csv files in this dataset:

- games.csv : all games from 2004 season to last update with the date, teams and some details like number of points, etc.
- games_details.csv : details of games dataset, all statistics of players for a given game
- players.csv : players details (name)
- ranking.csv : ranking of NBA given a day (split into west and east on CONFERENCE column
- teams.csv : all teams of NBA

3.2 Non-relational Dataset

Indonesia's Top E-Commerce Tweets

This dataset contains the tweets from the first tweet until April 2020 of top e-commerce unicorn in Indonesia namely Shopee, Tokopedia, Bukalapak, Lazada and Blibli.

- bliblidotcom.json: contains a series of tweets from "Blibli.com", discussing various topics like Ramadan, travel photography, and Labor Day, among others.
- bukalapak.json: contains multiple records detailing tweets from the "bukalapak" Twitter account, with each entry providing information on the tweet's content, creation date, engagement metrics (e.g., retweets, likes), associated media, and user interactions.
- lazadaID.json: contains a series of Twitter data from the user "lazadaid" ("Lazada"), detailing their interactions, mentions, tweets, and related meta-information for each tweet.
- ShopeeID.json: contains a series of tweets from the user "Shopee Indonesia" with various replies to other Twitter users, primarily consisting of friendly engagements and customer support interactions.
- tokopedia.json: contains multiple Twitter post entries from the user "Tokopedia" with details such as tweet content, date, time, hashtags, links, number of likes, retweets, and other metadata for each tweet.

Challenges:

Through research, we found several methods for loading datasets without loading the entire database into the memory. And we are trying to find the fastest way to load the data. We can read the data in chunks or use generators to read one row of data at a time. There is also a method involving Memory-Mapped Files. This method allows us to access the file as if it were in memory, but the OS manages the paging of data in and out of memory. However, this method requires us to use an outside library like mmap, which we are not sure if it’s allowed in our project.

Since we are not allowed to the entire database or table into the memory, we find it can be a bit tricky to join two CSV files or two JSON files together. We realize that a Nested Loop Join can be used to join two CSV/JSON files together. However, for this project, we assume a large amount of data for our databases and Nested Loop Join can be time-consuming/inefficient for large files. Therefore, we need to find an efficient way to load the data. There are a few other methods we are considering: sort-merge join, hash Join, and map-reduce.

For the query language in our interactive command line interface, we have a hard time deciding whether we should use the same query language for both relational databases and non-relational databases or two different query languages for relational and non-relational databases.