ADVANCED DATABASE
Group Journal
Team KUGA

Day 1
Date and Time: 29th March 11:00 am to 4:pm

Today we started our final project, the implementation of a simple database. We are going for a database that caters to generalised data, for which the database performs all CRUD operations. Our database works with query inputs from the user. We first started off by deciding what kind of file we would like to store on the disk. We decided to work with CSV files since we have worked on them in the past. We started off to get the simple stuff working first today. We successfully implemented a tokenizer for the query and the input of data into a file. We also successfully implemented the create table functionality and the insert records functionality. Our database takes input such as create table <table_name> (id,name,age). This will create an entire csv file stored on the disk with the headers id, name, and age in the file. When we use the query
Insert in <table_name> (1, Kushal, 20); this data is stored in the csv file. We then had to figure out the storage and indexing methods we wanted to use.

Our plan is to work with a single CSV file at a time. We create a hash table for each column with a pair to the ID value, which is stored in the code while we are working on a specific CSV file. After which, we will create a B tree structure of the Hash values that will allow us to access data faster. Although, even if one record is updated or deleted, we would have to reorder the B tree, which seems complicated, so we are still not sure of it. But once we come up with a better idea for this functionality, we can move ahead and implement the delete and update action.

Day 2
Date and Time : April 5th, 9:00 am to 2:00 pm

In today's work, we are continuing from where we left off, and we tried to implement the update and delete functionality. We were able to implement both functions partially. The update function would take queries such as *"update <csv_table_name> <column_name> <new_value> where <condition_column> = <condition_value>"* and the delete function would take queries such as *"delete from <csv_table_name> where <condition_column> = <condition_value>"*. Although both functions were not able to reflect the changes in the index files,. Updating would just add another record with the changed information. Delete would not do anything at all in the index files. Both functions worked perfectly in the main csv file but not in index files. We started to realise that creating an index file and then indexing a b tree in it would be a complicated and memory intensive task.

Day 3
Date and Time: 11th April, 12:00 pm to 6:00 pm

We had a talk with our professor and changed our approach. We started a new code that just works around the B tree. Each column is made into a b tree, and all the crud operations are performed on it. When wanting to store the table, we can store the data in a csv file. We are not making index files anymore. Since we have a B tree for each column and a general database, we are able to search for data even if the specific column is not mentioned. The code will traverse through each column (b tree) and, after that, return the corresponding record values. All the crud operations are working, and we have our indexing and storage mechanisms in place. The only changes we plan to make are to implement the similar query style we had initially in the form of user statements. Right now, we are mentioning in the code what actions it should take and that it is just working with the class call function.

Day 4
Date and Time : 13th April, 5:00pm to 9:00pm

On day 4, our team made significant strides in advancing our database management system (DBMS) project. With the successful implementation of the B-tree structure, we transitioned our focus towards incorporating all CRUD operations into our system. Collaboratively, we integrated SQL queries from our previous codebase, refining functionalities such as update, delete, search, and insert. Each team member played a crucial role in this process, with individuals taking ownership of specific tasks. In particular, attention was devoted to refining the delete, search, import, and select functions to ensure their seamless integration and efficient performance within our DBMS. Discussions were held to determine the optimal structure and formatting of queries, ensuring user-friendly interaction with the system. With a collective effort, we ensured that the system provided clear prompts for users to input queries, enhancing usability.

Moreover, the implementation of the import function marked a significant milestone for our project, enabling the seamless integration of external data sources into our system. This feature automatically generated B-trees for imported columns, streamlining subsequent CRUD operations. Additionally, the development of a display function provided users with comprehensive visibility into all changes made within the system.

Furthermore, to enhance user experience, we incorporated functionality to automatically print column headings, eliminating the need for users to repeatedly enter ".schema". This attention to detail reflects our commitment to optimizing usability and efficiency within the system.

In addition to these advancements, we also implemented an export function, allowing users to export data to a CSV file. Notably, changes made within the system are first reflected in the B-tree structure. Only when the user inputs the export query does the CSV file get updated, ensuring a streamlined and controlled process for data export.

Overall, day 4 was characterized by productive collaboration, successful implementation of key features, and rigorous testing to ensure the robustness and reliability of our DBMS. As a team, we are pleased with the progress made and excited about the potential of our project moving forward.