2021

Feasibility Study

RDBMS – DP 25

DHRUMIL RAKESH SHAH
dh647095@dal.ca

MANRAJ SINGH
mn697903@dal.ca

VISHVESH NAIK
vishvesh@dal.ca

Meeting Details:

- Meeting 1: 03-06-2021
    - Discussed Agendas:
        - Introduced ourselves to get to know each other and get comfortable with each other as a team.

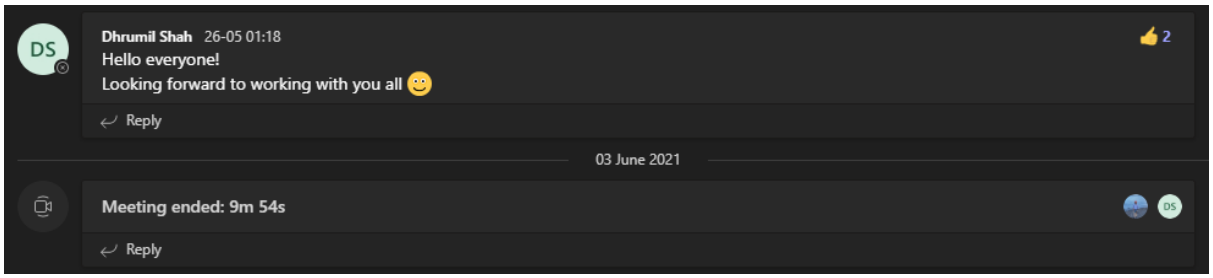

Figure 1: Meeting 1

- Meeting 2: 05-06-2021
    - Discussed Agendas:
        - We discussed the Relational Database and RBDMS.
        - We discussed the flow of the application.

Figure 2: Meeting 2

- Meeting 3: 08-06-2021
    - Discussed Agendas:
        - In this meeting, we discussed the various data structures that can be used for our application.
        - Discussed how to store the data.
        - Also discussed the end-to-end flow of the application.

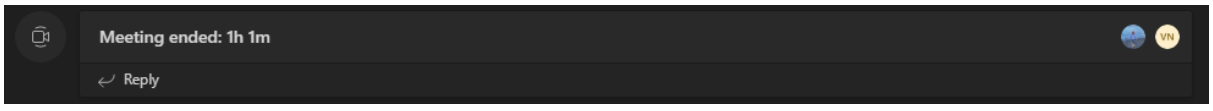

Figure 3: Meeting 3

- Meeting 4: 09-06-2021
    - Discussed Agendas:
        - In this meeting, we discussed the design features to be implemented.
        - We discussed the ACID properties and Gantt chart as per our project requirements.
        - We also discussed how we will analyze the Queries.
        - We divided the report's work among ourselves with everyone's consent.

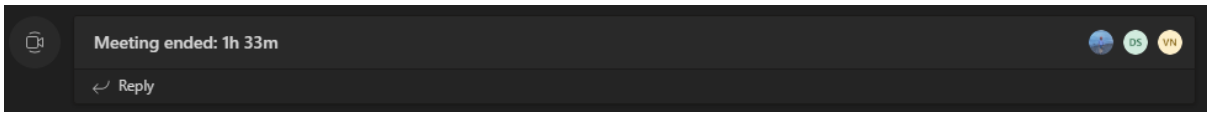

Figure 4: Meeting 4

- Meeting 5: 09-06-2021
  - Discussed Agendas:
    - We started with checking up on the progress done for the assigned tasks in the previous meeting.
    - We discussed the Flow chart, Algorithms, and Pseudocode describing the workflow of the application.
    - We also discussed the initial Test Cases that needed to be performed.
    - We finalized upon the minute details required for the report and the application.
    - We also discussed how the query execution engine will function.

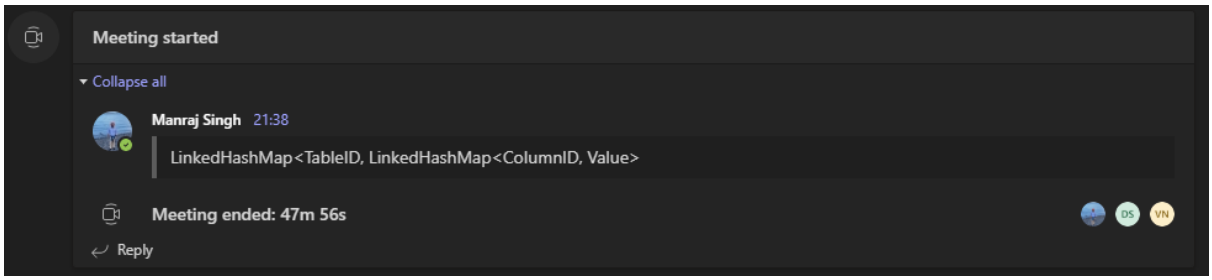

Figure 5: Meeting 5

- Meeting 6: 10-06-2021
  - Discussed Agendas:
    - We started with checking up on the progress done for the assigned tasks in the previous meeting for the report.
    - Gave suggestions for each-others work.
    - Collaborated on the work done by each team member and formatted the report.

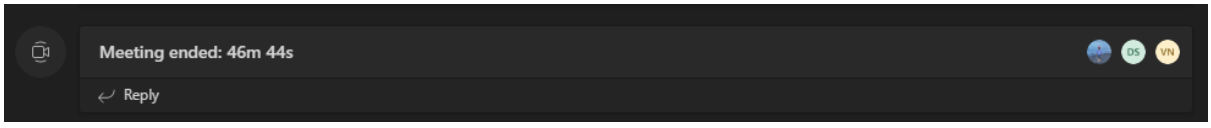

Figure 6: Meeting 6

Framework and Related Technologies:

- **Programming Language** – JAVA
  - Java classes are rich and support user-defined data types.

- Methods of Java classes provide new functions in SQL.
- Java also provides collections such as Map, List which can be used to store or retrieve data.
- Java also supports a relational database management system and is platform-independent.

Data Structure:

For this project, we will use **Linked HashMap**, **ArrayList**, **Stack,** and **Queue** data structures. Linked HashMap is implemented using the Map interface. We chose this data structure because of its ease of use and various advantages. It is a combination of HashMap and List data structure utilizing the properties of both. Time Complexity of finding any value using a key is O(1), and also the nodes are added serially to be fetched in the same order, which is not possible in a standard HashMap.

Linked HashMap node example:

LinkedHashMapEntry<key, value> previousInput;

LinkedHashMapEntry<key, value> nextInput;

Object keyObj;

Object valueObj;

HashMapEntry<key, value> nextEntrySameIndex;

As we can observe from above, each Linked HashMap is an Array of nodes where an individual node consists of its previous and next node pointer and contains the current node's key and value objects HashMap Entry for repeated hashcode index but separate keys.

Each Linked HashMap also consists of an algorithm to find the index and hashCode of the node in the array of nodes, calculated using the key. In an ideal scenario, the algorithm produces a unique index and hashcode for every key, but in reality, there are chances of hash Collision for which the list structure of hashMap is used in which each node of the hashmap can have the next node pointer referencing to the node with the same index but different key. If the hash collision happens due to the same key, then the value of that key is overwritten, and the previous value is returned with the put() method call.

There are also chances when the node's array length exceeds, then the array's length is increased by double, and all nodes are moved from the old array to the new array. This operation has a time complexity of O(n).

Stack is the data structure that stores and retrieves data in a LIFO (Last in First Out) manner. And Queue is the data structure that stores and retrieves data in the FIFO (First in First out) manner. Both are mainly used for logging purposes.

Assumptions:

- Transactions will not interact with multiple tables at a time, i.e., Joins will not be used.
- Only one query will execute at a time, i.e., there will be no nested queries.
- No Alias will be used in the query.
- Queries will not have a GROUP BY clause.
- Simultaneous query execution is not allowed, i.e., the user can run a new query after the first one finishes execution.
- Query results will be limited to 1000 records at a time.

Flowchart:

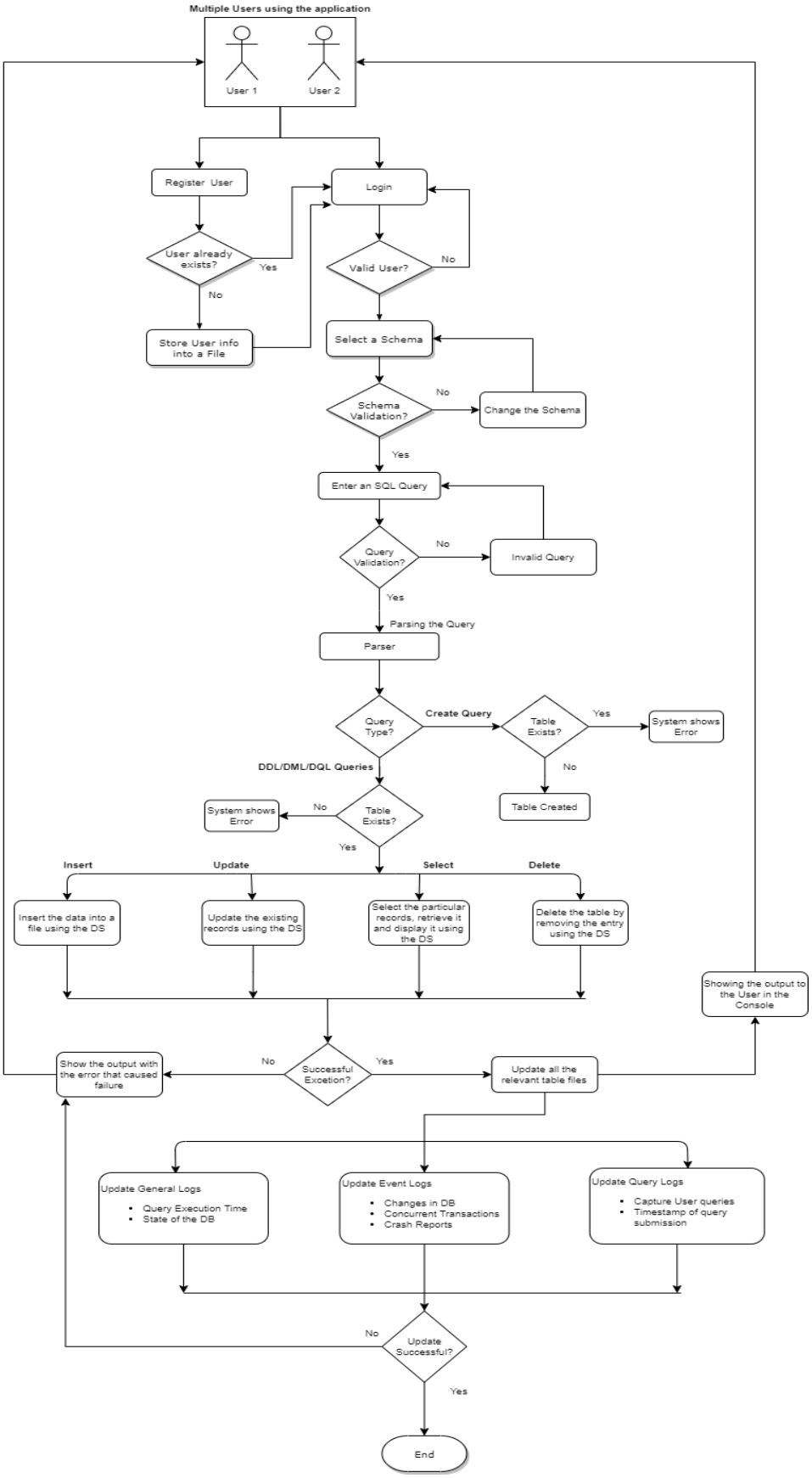

Figure 7: Flowchart of the custom RDBMS

Algorithm:

1. Start the application
2. Register User
   2.1. If the user already exists, go to step 3.
   2.2. If the user does not exist, then the user registers.
   2.3. Storing the user information into a file, go to step 3.
3. User Login
   3.1. If the user credentials are valid, go to step 4.
   3.2. If the user credentials are not valid, go back to step 3.
4. Selecting the Schema
   4.1. If Schema validates successfully, go to step 5.
   4.2. If Schema is not validated, then Change the Schema, go to step 4.
5. User enters an SQL query
   5.1. If the query is valid, go to step 6.
   5.2. If the query is not valid, go to step 5.
6. The Parser parses the query
   6.1. If it is a Create query
      6.1.1. If the Table exists, go to step 6.1.2.
      6.1.2. Show error to the user that the Table already exists.
      6.1.3. If the Table does not exist, go to step 6.1.4.
      6.1.4. Create a new table.
   6.2. If it is an Insert query
      6.2.1. If the Table exists, go to step 7.
      6.2.2. If the Table does not exist, go to step 6.2.3.
      6.2.3. Show error to the user that the Table does not exist.
   6.3. If it is a Select query
      6.3.1. If the Table exists, go to step 7.
      6.3.2. If the Table does not exist, go to step 6.3.3.
      6.3.3. Show error to the user that the Table does not exist.
   6.4. If it is a Delete query
      6.4.1. If the Table exists, go to step 7.
      6.4.2. If the Table does not exist, go to step 6.4.3.
      6.4.3. Show error to the user that the Table does not exist.
7. Execute the query
      7.1.1. If query execution is successful, go to step 8.
      7.1.2. If query execution failed, go to step 7.1.3.
      7.1.3. Show error to the user that the query failed to execute.
8. Update the relevant tables in the DB.
9. Update General Logs
   9.1. If updated, go to step 12.
   9.2. If not updated, go to step 9.3.
   9.3. Show error to the user.
10. Update Event Logs
   10.1. If updated, go to step 12.
   10.2. If not updated, go to step 10.3.
   10.3. Show error to the user.
11. Update Query Logs
   11.1. If updated, go to step 12.
   11.2. If not updated, go to step 11.3.

11.3. Show error to the user.
12. Update successful
    12.1. If the user wishes to continue
    12.2. If yes, go to step 5 and for different Schema, go to step 4.
    12.3. If no, go to step 13.
13. End the user session by exiting the application.

Design Features:

- In this project, we are building a simple relational database and its database management system.
- This system will handle multi-user (2 users) requests.
- The system will be used on the command-line interface (CLI) and will perform the required functionalities of a relational database management system (RDBMS).
- The database will implement the ACID properties.
- This system will follow concurrency control and apply two-phased locking.
- The system will store General Logs like query execution time, storing the state of the DB.
- It also stores the Event Logs like what all changes occurred in the DB and crash reports.
- The system captures the user queries and the timestamps of their execution through the Query Logs.
- SQL Dump can also be created with the system.
- The system will also create a simple ERD diagram from the existing DBs based on their structures.

Implementation Plan:

Input Validation:

- The system will first perform input validations for Login and Register functionalities.
- The system will check if any of the Schema is selected.
- The system will initially parse the SQL statement that is written to check if it is from;
    - Create
    - Insert
    - Update
    - Delete
    - Alter
    - Use statement
    - Drop
- A pre-defined function of the String class will be used .contains() to determine the query's functionality.
- If the query does not have any such components (keywords), an error will be thrown.
- Specific and precise validations can be done by using the loops in Java.
- RegEx can be used to validate the query structure that is being given as input to the system.
- We can also use the split() function to differentiate the functionalities of the query.

- After all, this done, the values are pushed onto the Stack, as it follows the LIFO structure;
    - Validation is done for the brackets being closed after we find an open bracket.
    - Validation is done to check if the number of columns and the retrieved values are equal or not.
    - After completely parsing the query, Stack should be empty, so if anything is remaining in the Stack means that the query is invalid.
    - The main thing to validate using the Stack is to check whether the query is balanced or not.

Boundary Cases Validation:

- After query validation successfully completes, boundary cases are considered.
- The Schema and Table names will be validated. The system checks the directory looking for the Schema and the respective Table. If not found, then an error is shown.
- After the Table is being successfully identified, the system will read the Metadata file.
- The validation is done for column names, type, and size (parameter size) after validating the Table.
- Query values will then be parsed using a Parser based on the metadata.
    - Example integer should not be able to store text/varchar values.
- The primary key is validated, i.e., no duplicates are allowed.
- After that, the foreign key is validated based on its constraint. The value should be NULL or present in some other table that is linked to this Table.
- Deletions won't be completed if there are dependency constraints like a foreign key constraint, need to delete the Table whose foreign key is present in the current Table.
- Rollback and commit operations are also validated, and various logs being recorded and written in the files.

Backend Operations:

Database:

- The data in our code will be stored in ArrayList<LinkedHashMap<Key, Value>> format.
- Internal Linked HashMap will store the database column name as the key and the column row value for that row as the value.
- Only one row will be pushed to the Linked HashMap, and the whole ArrayList will represent the Table.
- A separate HashMap will be maintained to keep track of all the databases created. Its key-value pair will be the name of the database and the ArrayList reference.
- While storing the rows, if the columns are limited, then the other column values will be set null.

DBMS:

- Users can signup and login into the database. His details will be stored and fetched from a HashMap maintained with the user details.
- When the user login to the database, it will generate a thread for the user.

- After validating the query, its timestamp will be stored as query logs along with the query.
- State Logs will also be updated after every query execution with that query timestamp.
- Crash details and the exceptions will be printed in the event logs and other state details of the database.
- A stack will be maintained in the database to store values like name, row, column, the previous value, current value of the database Event Stack.
- If the user rolls back or is inactive for long, the tables will be reverted back to the last commit using Event Stack.

ACID Properties:

- **Atomicity:** Any update in the Table will be made after the commit command. If there will be any exception while updating, the data will be rolled back.
- **Consistency:** We will keep track of the database state after each transaction; when trying to delete a table, all the foreign keys referencing the Table's primary key will be deleted first.
- **Isolation:** A separate HashMap will be maintained to track the table availability. So that no two users can change the Table at the same time.
- **Durability:** After a successful commit, the data will be visible to all the users. Even in the case of an exception, the data will still be preserved.

Storage:

- The properties file of the system will store the Root directory, which will eventually store the Schemas and the Tables themselves.
- Individual folders under the Root directory will represent a Schema, whereas individual tables can be represented by a single folder specifically for that Table. There will also be an Event Logs table within the Schema.

Initial Test Cases [Unit Tests]:

User Login:

- Check for the username and password validation
- Multiple logins should be checked.
- Check if the user can log in into the DB using valid credentials,
- Check if the user cannot log in into the DB using invalid credentials,

SQL Queries and ACID properties:

- Basic queries should execute successfully like:
    - Select
    - Update
    - Delete
    - Insert
    - Create
    - Alter

  - Use statement
  - Drop
- Check whether the queries adhere to the correct writing format.
- Table name and schema names should be validated.
- Exceptions and errors should be shown when an invalid query is executed.
- Queries should be validated like: Table names and Schema names should be present in a query.
- Check if the data is updated into DB when the query execution is done.
- Data should be reverted to the original state or previous state in case of execution is not complete.
- Check the Table locking so that no multiple users can update into the same Table.
- Table drop should not impact other tables where its primary key is used as a foreign key.
- Check if the SQL dumps are being created.
- Check if the logs are being created.

Gantt Chart:

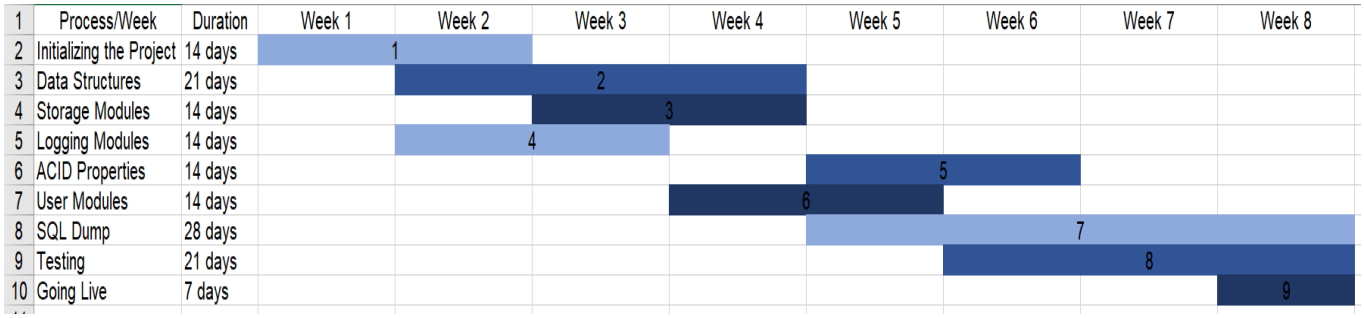

Figure 8: Gantt Chart