



Faculty of Computer Science

Final Report

Final Project

Group 10

Rashmi Chandy

Amelia Itzel Hernandez Beltran

Kethan Kumar Nasapu

Submitted to

Dr. Saurabh Dey

CSCI 5408 Data Management, Warehousing, Analytics

Halifax, NS, December 2, 2020

Table of contents

1. Introduction	2
2. Conceptual Framework	3
2.5 Flowchart	5
3. Implementation	6
3.1 Pseudocode	9
3.2 Possible test cases	9
4. Team Members	11
4.1 Amelia	11
4.2 Rashmi	11
4.3 Kethan	11
5. Product Features	12
6. Conclusions	14
7. Future work	14
8. References	14

1. Introduction

The final project of the course CSCI 5408 consists in the development a Database Management System (DBMS). The programming framework has to be designed to support the fundamental functionalities offered in DBMS applications on the market like MySQL or Oracle. These functionalities include the creation of users, databases, and tables. The DBMS also performs CRUD operations: select, insert, update, and delete. Likewise, it handles multiuser requests, i.e., two users can perform the CRUD operations without violating the ACID properties. Moreover, the system can generate a simple SQL dump with the tables structures of a particular database. The system also offers the possibility to create a simplified ERD diagram that shows the entities, the relationships between the entities, and the foreign keys that connect them. Users can interact with the system through SQL queries, which are parsed, validated and then processed to execute the necessary action on the data stored on the databases. General and Event logs are captured to record the queries performed on each database and the time where they occurred.

2. Conceptual Framework

2.1 Language used in implementation of DBMS:

Python as the main programming language because it offers easiness for handling data

2.2 Problem Statement:

The objective of this project is to create a database management system that will handle capabilities of:

- Processing database queries
- Data storage
- Data retrieval
- Building a database
- Log analysis
- Handling multi-user requests
- Providing a command line interface
- Maintaining a Data Dictionary
- Supports Reverse Engineering to create Entity Relationship Diagram

2.3 Flow of Application:

Algorithm:

1. Start application
2. Display menu options and prompt user for input
 - a. New user
 - b. Existing user
 - c. MySQL Command prompt
 - d. MySQL Dump
 - e. Export ERD
 - f. Exit from application
3. If option a:
 1. Input username and password of new user
 2. Encrypt newly created user's password
 3. Check if username already exists
 4. Add username and encrypted password to database storage file
 5. If INVALID
 - Display error message to user
 - Else
 - User created successfully
- If option b:
 1. Input username and password of existing user
 2. Decrypt the user database file
 3. Validate user password
 4. If INVALID
 - Display error message to user
 - Else

Login successful

If option c:

1. Load MYSQL command prompt
2. Enter SQL query – CREATE DATABASE/ USE DATABASE/ INSERT/ SELECT/ UPDATE/ DELETE
3. Validate SQL Query structure to check if query in standard SQL format
4. Parse SQL Query (to dictionary format)
5. Execute SQL Query by using parsed data values
6. Save in database (.csv file)
7. If INVALID

Display error message to user

Else

Query Execution Successful

8. While user is logged: Step 1 can be repeated which allows user to enter new queries

If option d:

1. Input the username
2. Input database name
3. Input file path for SQL dump
4. Check If database present
5. If Database present
 - Create Dictionary with table name, attribute, data type, primary key, foreign key, table relationship, attribute relationship
 - Create csv file with the generated SQL dump data in the required file path

Else

Display error message to user

If option e:

1. Input the file path of the table
2. Check if given table present in database
3. Using data dictionary, it will create an ERD of the table structure

2.4 Data Structures:

Our implementation uses various data structures to store and extract the necessary data and metadata of our system. Data structures used are:

Data Frame

A data frame is a two-dimension data structure that, by nature, is organized in a row-column format; therefore, it is ideal for storing the table's content. Data frames can be easily indexed for selecting some or all rows and columns of data present in it. Operations such as inserting, deleting, updating can be performed efficiently. Data Frame is widely used in our application for storing data in our database

Dictionary

Dictionary is useful data structure which can hold a collection of values. It facilitates faster retrieval of data by referencing its key. Hence it is useful while parsing values inside a table. Dictionaries have the

capability to update or delete key value pairs hence this mutable nature will be well suited to fulfill the needs of the database management system.

2.5 Flowchart

The flow we will follow for the DBMS development involves several steps. First, we will ask the user for its username and password. The user will type it in the command line. Second, we will perform the credentials verification to see if the user has access to the DBMS. If the verification is not successful, the program will show an error indicating it. Third, if the user was successfully identified, the DBMS will recover the last state of the system for that user, and now the user can type a query to access a database. After the user enters the command, the program will read the input stream. Fourth, the program will verify if the query is valid. If it is not, it will display an error message to the console to tell the user about it. Fifth, if the query is valid, the program will start the parsing of the query. Sixth, with the query elements, we can now know what action we need to perform in a database and call the most appropriate function to handle the create, read, update, or delete operation. In this step of the process, we will also update the logs containing the DBMS status. Finally, we will display a message if the query was successful or if there was a problem.

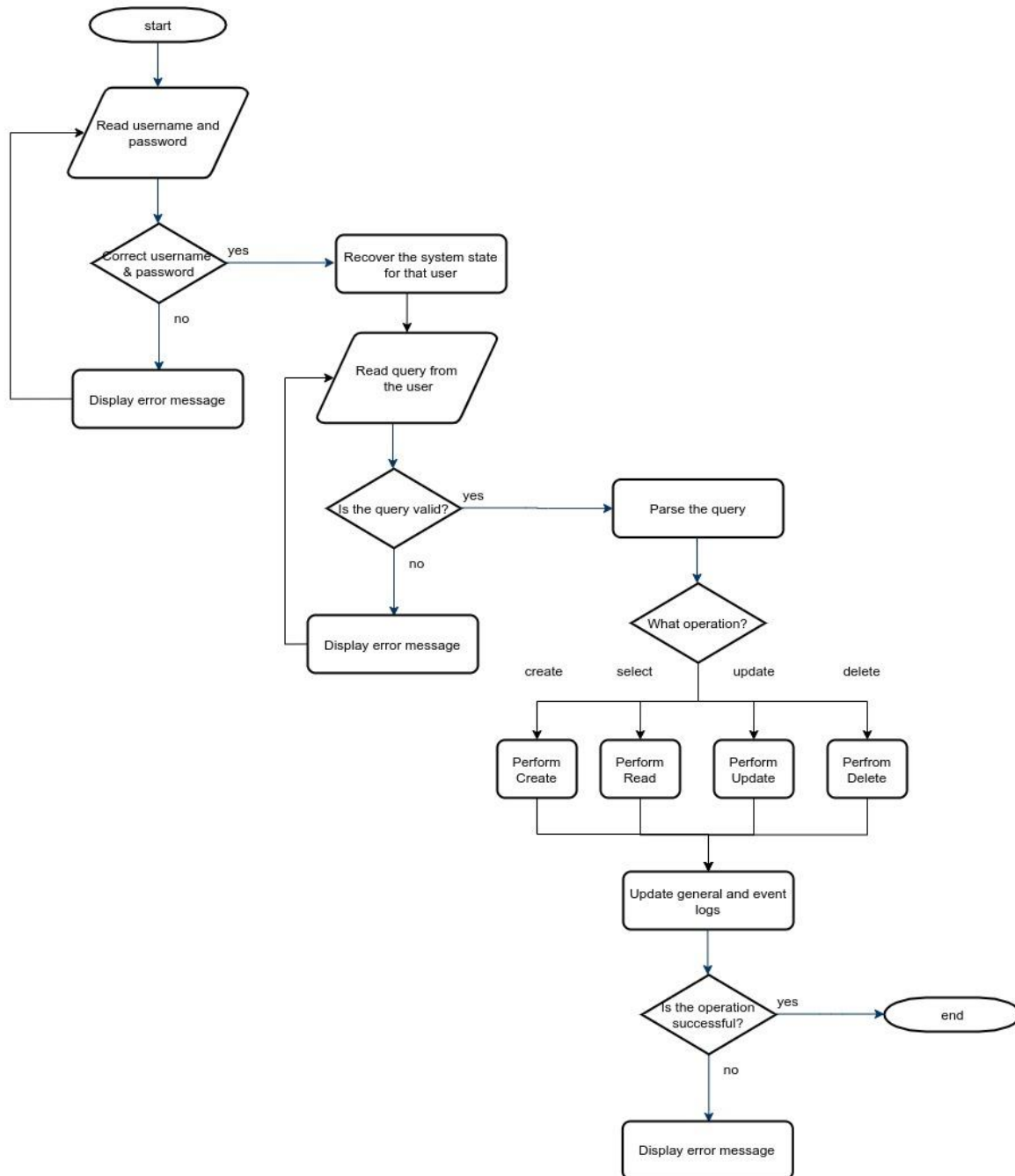


Figure 1. Flowchart

3. Implementation

For the implementation of the entire DBMS system we used the programming language Python because it offers different useful tools to handle data. In specific, for handling the data in the database tables, we used the library Pandas which offers the ability to access CSV files in an easier way. It is important to

mention that for the implementation of the DBMS in this project, we did not use any additional third-party library that already implements the work of a DBMS. All the methods were designed and developed from the members of the team

Users

The user related data such as username and password are encrypted and also stored for later access. As part of the implementation, we allow users to register and select a username as password. These values are stored so that the users can access their databases again. When a user wants to access a database, it will be required to provide that username and password.

SQL validation

The SQL query is validated to check if the query which user enters matches the standard SQL query structure. Each part of the SQL statement is validated against the regex expression and Regex expression can be used to match parts of the query from which data can be extracted and passed to the parsing section.

SQL parsing

The SQL parsing is implemented using regular expressions to extract the reserved words that indicate a specific query. The method we developed for parsing also extracts the necessary values in the query, e.g., the values to insert on a table.

Storage

We decided to use CSV files for the storage of the tables' data, data dictionary, and any other necessary file in the system. This kind of files are an appropriate option because they offer a structure that is similar to a table so that it is easy to understand the content's relationships. The Pandas library offered us the possibility to access the CSV files in an easier way and then manipulate the content as a data frame.

CRUD operations

For the CRUD operations, we first created a controller which identified the query to interpret. The controller called the appropriate method to process the query and do all the necessary validations. As we mentioned before, the basic operations available in the DBMS are select, insert, update, delete, create database, select database, and create table.

Concurrency control

The concurrency control mechanism is essential in a DBMS in order to maintain the ACID properties. These properties are an indicator that a database works properly. For implementing the concurrency control, we did a table level lock, i.e., when a user is manipulating a certain table, another user needs to wait until is liberated. To achieve this, before doing any change in a table, the system checks if the table is in use. The system knows if a table is in use because it has access to a file called *status* and in this file there is a flag that indicates if the table is in use or not. In order to avoid a dead lock in the system. A user A only waits for a certain amount of time (18 seconds) for the table to be release by user B. If after

that amount of time, the table is not released, then the user need to try again by entering the query another time.

Data Dictionary

The DBMS has a data dictionary that stores relevant details about the tables in a database such as attribute, data type, low limit, upper limit, primary key, foreign key, relationship table, and relationship attribute. The data dictionary is a key element in the design of the DBMS because before executing a query, we need to access it and see the relationships between tables and attributes to make the appropriate change. An example can be, if the DBMS stores two tables, e.g., the customers (parent) and orders (child), it is not possible to first store a value in orders if a value is not present in the customers table; therefore, the system need to learn about those relationships in the data dictionary.

SQL dumps

For the SQL dump, the system accesses the data dictionary to get structure of the tables. As a result, the SQL dump retrieves a CSV file with the structure of all the tables present in a database. This will help users to learn about the attributes of their tables, data types, and relationship. All is present in a single file. To create a SQL dump, it is necessary to provide the path and filename to store the file with the dumps.

ERD

For the ERD, the system also accesses the data dictionary to get structure of the tables. As a result, the ERD query provides a file with the relationships between tables with the following format Table1 -> Table2 | KeyTable1 – KeyTable2. Instead of Table the DBMS provides the entity names and instead of KeyTable the DBMS provides the name of the attributes that connect both tables.

Log files

For the general and the event logs, we used LOG files with different logging levels such as warnings, errors, or info. These levels are shown in form of a tag with its corresponding timestamp, which provides clarity for the user that intends to use and understand the logs. Information related to the queries, execution time and state of the database are present in these files so they can be easily referenced in case of any failure in the system.

Data structures

As mentioned before, we used various data structures like lists, dictionaries, and data frames. These kind of data structures are ideal for handling data and make it easier to access values that come from the SQL queries and then store them as a CSV file in the system.

Code architecture

For the code, we used a modular structure, i.e., we created methods to handle each of the different operations for the DBMS. For example, the SQL parsing, create, read, update, and delete operations have their own methods. With this code structure, it is easier to update the code later or add more features if necessary.

3.1 Pseudocode

Below, we present the pseudocode logic used to develop the DBMS. This logic is based on the data structures and the flowchart we previously described.

```

BEGIN
  READ username and password
  VALIDATE if username and password are correct
  IF valid
    PRINT a menu with options (sql prompt, export sql dump, export erd, logout)
    WAIT for input with an option from the menu
    READ the string that the option
    IF sql prompt
      PARSE the query
      VALIDATE if the string is a legitim sql query
      IF valid
        SWITCH
          Case create database: perform create database operation
            UPDATE data dictionary
          Case select database: perform select database operation
          Case create table: perform create table operation
            UPDATE data dictionary
          Case select: perform read operation
          Case insert: perform insert operation
            RUN concurrency control mechanism
          Case update: perform update operation
            RUN concurrency control mechanism
          Case delete: perform delete operation
            RUN concurrency control mechanism
          Case sql dump: perform export sql dump operation
            READ data dictionary
          Case erd: perform export erd operation
            READ data dictionary
        UPDATE general log
        UPDATE event log
      ELSE
        RETURN "error message"
  END

```

3.2 Test cases

For testing the application, we have considered a couple of test cases to validate the correctness of the program.

- Validating the credentials of the user to implement access control
- Checking if all supported queries are parsed correctly.
- Check the tables to see if the values are updated correctly.
- Check if the data dictionary is updated correctly.
- Check if logging is performed after each query execution.
- Test the CRUD operations to ensure that the table output file is updated as expected
- Check if user is able to export the SQL Dump
- Check functionality if an ERD file can be created for an existing database.

For performing the previous test cases, we used a simple example of a database that includes the following SQL queries.

```
CREATE DATABASE sampledb;
```

```
Use sampledb;
```

```
CREATE TABLE customers(  
  customerNumber int,  
  customerName varchar,  
  customerLastname varchar,  
  city varchar,  
  stalCode int,  
  PRIMARY KEY (customerNumber),  
);
```

```
CREATE TABLE orders (  
  orderNumber int,  
  orderDate varchar,  
  shippedDate varchar,  
  status varchar,  
  customerNumber int,  
  PRIMARY KEY (orderNumber),  
  FOREIGN KEY (customerNumber) REFERENCES customers (customerNumber)  
);
```

```
CREATE TABLE orderDetails(  
  orderNumber int,  
  productCode varchar,  
  quantityOrdered int,  
  priceEach decimal,  
  PRIMARY KEY (orderNumber),  
  FOREIGN KEY (orderNumber) REFERENCES orderDetails (orders)  
);
```

```
SELECT * from customers;
```

```
INSERT into customers(customerNumber, customerName, customerLastName, city, postalCode)
values(2, 'Kevin','Kwan','Ontario', 'B3H 4R2' );
```

```
UPDATE customers SET customerName = 'Diana'
WHERE customerNumber = 1
```

```
DELETE FROM payments WHERE customerNumber = 1;
```

4. Team Members

4.1 Amelia

Roles: SQL Execution: Create database/CREATE/INSERT/UPDATE/SELECT/DELETE, SQL Dump, ERD, Data dictionary, Concurrency Control

New Learnings: I learned in more depth how databases work because the level of understanding is entirely different when you use a system like MySQL than when you try to implement it, even when it is only with the most basic functionalities. Before this project and this course, I did not know very much from databases like a SQL Dump, a Data Dictionary, the concurrency control mechanism, but after implementing them, I feel I understand their relevance and their use.

Difficulties: The most difficult part in this project was related to the development of the concurrency control mechanism because it requires a good understanding of the database works. Likewise, it requires good programming abilities for implementing it in an appropriate way.

4.2 Rashmi

Roles: SQL Validation and Parsing of: CREATE/SELECT/UPDATE/CREATE DATABASE/USE SCHEMA, Logging

New Learnings: Creating a database from scratch has helped me to understand fundamental database management design and the core components that make up its implementation. I was able to understand the flow of execution that happens in the background when any query is submitted in any database management system. By working on SQL parsing and validation, I realized its importance and was able to learn how to implement it using basic regex validation.

Difficulties: It was difficult to work with Data frames in python as it was very new to me.

4.3 Kethan

Roles: User creation, authentication and management, Logging, SQL Validation and Parsing of: INSERT/DELETE, Concurrency Control

New Learnings: I gained deep insight into creating a database and its management system by implementing various concepts of query parsing, concurrency control and logging. Additionally, I was able to understand the DBMS workflow through this practical implementation. The concurrency control

mechanism helped me to understand locking mechanisms. I was able to get a clear picture of ACID properties and understand its importance to a greater extent.

Difficulties: It was my first time working with pandas library and initially it was difficult to implement conceptual ideas using the data frame.

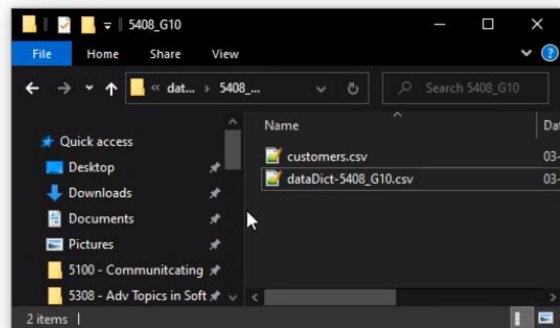
5. Product Features

Below, we can see some screenshots that show some of the functionality of the program using different SQL queries.

```
C:\Python38\python.exe "D:/Dal/Academics/Fall 2020/5408 - Data Mgmt, Warhsng Analytics/Project/5408-finalproject/kethan.c
[***** Welcome to Custom My SQL Database *****]
1. New user
2. Existing user
3. MySQL Command prompt
4. MySQL Dump
5. Export ERD
6. Exit from application
Select one of option from above:3
```

```
***** Welcome to My SQL Command Prompt *****
Command to exit command:logout
SQL>CREATE DATABASE 5408_G10
---The query is create database---
Database 5408_G10 created
SQL>
SQL>use 5408_G10
---The query is use---
Use 5408_G10
```

```
***** Welcome to My SQL Command Prompt *****
Command to exit command:logout
SQL>CREATE DATABASE 5408_G10
---The query is create database---
Database 5408_G10 created
SQL>
SQL>
SQL>
SQL>USE sampledb
---The query is use---
Use sampledb
SQL>
SQL>
SQL>use 5408_G10
---The query is use---
Use 5408_G10
SQL>
SQL>
SQL>CREATE TABLE customers (customerNumber int,customerName varchar,customerLastName varchar,city varchar,postalCode int)
---The query is create---
Table created
SQL>SQL>
```

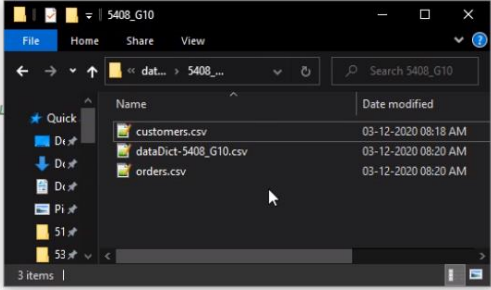


```

SQL>use 5408_G10
---The query is use---
Use 5408_G10
SQL>
SQL>CREATE TABLE customers (customerNumber int,customerName varchar,customerLast
---The query is create---
Table created
SQL>SQL>

SQL>SQL>
SQL>
SQL>
SQL>
SQL>
SQL>CREATE TABLE orders (orderNumber int,orderDate varchar,shippedDate varchar,status varchar,customerNumber int,PRIMARY KEY (orderNumber),FOREIGN KEY (custome
---The query is create---
Table created
SQL>SQL>

```

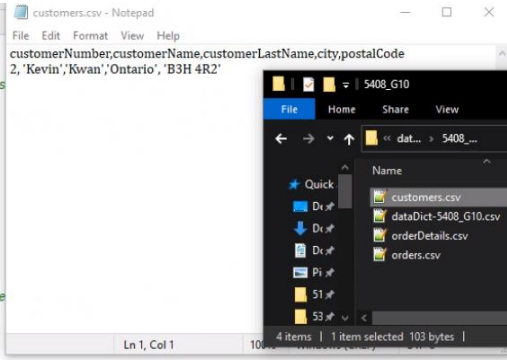
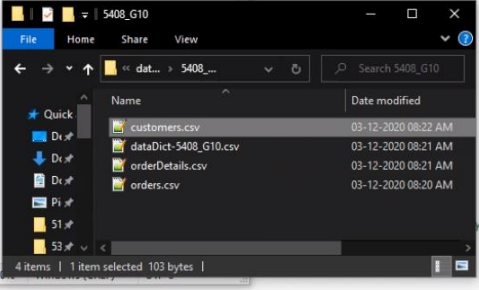


```

main
SQL>
SQL>
SQL>CREATE TABLE customers (customerNumber int,customerName varchar,customerLast
---The query is create---
Table created
SQL>SQL>

SQL>SQL>
SQL>
SQL>
SQL>
SQL>CREATE TABLE orders (orderNumber int,orderDate
---The query is create---
Table created
SQL>SQL>CREATE TABLE orderDetails (orderDetailsId int,orderNumber int,productCode varchar,quantityOrdered int,priceEach decimal,PRIMARY KEY (orderDetailsId),FO
---The query is create---
Table created
SQL>INSERT INTO customers (customerNumber,customerName,customerLastName,city,postalCode) VALUES (2, 'Kevin','Kwan','Ontario', 'B3H 4R2' );
---The query is insert---
SQL>

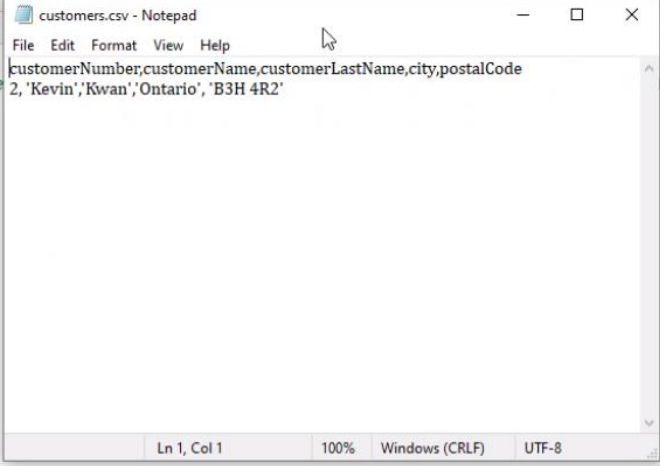
```

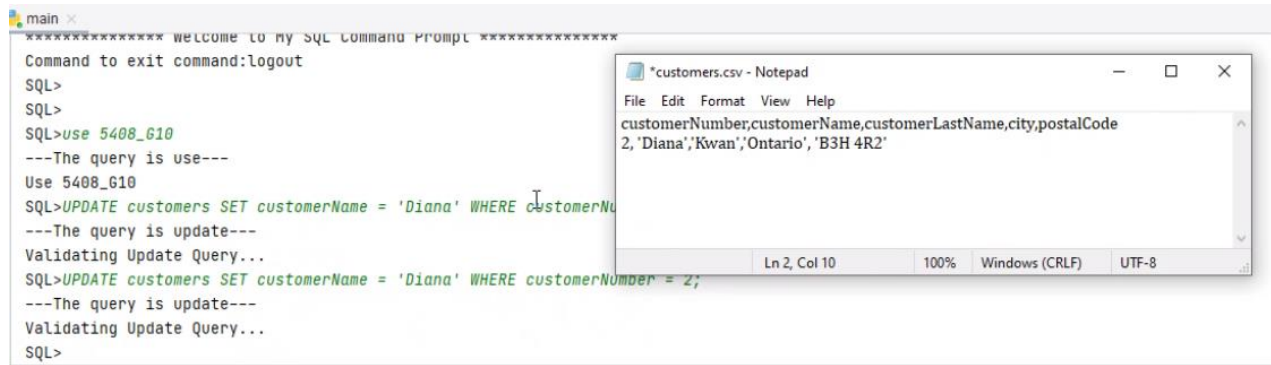



```

main
---The query is insert---
SQL>INSERT INTO orderDetails (orderDetailsId, order
---The query is insert---
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>SELECT * from customers;
---The query is select---
customerNumber customerName customerLastName city postalCode
0 2 'Kevin' 'Kwan' 'Ontario' 'B3H 4R2'
SQL>

```





The screenshot shows a Windows environment with two open windows. The background window is a SQL Command Prompt titled 'main'. It displays a welcome message and a series of commands and responses: 'Command to exit command:logout', 'SQL>', 'SQL>', 'SQL>use 5408_610', '---The query is use---', 'Use 5408_610', 'SQL>UPDATE customers SET customerName = 'Diana' WHERE customerNu', '---The query is update---', 'Validating Update Query...', 'SQL>UPDATE customers SET customerName = 'Diana' WHERE customerNumber = 2;', '---The query is update---', 'Validating Update Query...', and 'SQL>'. The foreground window is a Notepad application titled '*customers.csv - Notepad'. It shows a CSV file with the following content: 'customerNumber,customerName,customerLastName,city,postalCode', '2,'Diana','Kwan','Ontario','B3H 4R2'. The status bar at the bottom of the Notepad window indicates 'Ln 2, Col 10', '100%', 'Windows (CRLF)', and 'UTF-8'.

6. Conclusions

Thus, our implementation will act as a DBMS application layer and support the core DBMS functionalities such as data storage, retrieval, logging and visualization that are offered through a command line interface. The data structures that are selected to implement the database will be flexible to handle the I/O operations that are done by the user. The SQL queries will be validated and parsed in order to execute the actions that are expected by the user. Our database supports multi-user requests and manages access to the databases through user's credentials. In addition to this visualization of the data set is provided to the user in order to study and explore the data present in each table. This visual representation will help to understand the relationship between the entities in a database.

7. Future work

Currently our database can handle all basic queries, export SQL dumps, export a simplified version of an ERD; however, there are still many functionalities that can be improved. First, our system can only handle basic queries; therefore, it would be necessary to enhance it by adding the regular expressions that allow to parse a more varied types of queries. Second, the concurrency control mechanism implemented can be enhanced it to allow a smaller granularity. In the moment, the mechanism locks at the table level, which can reduce the performance of the database. Third, each of the methods that process and execute the CRUD operations can be optimize it to reduce the latency in the execution time. Fourt, the ERD file can also include a graphical interface or a diagram to make it more similar to an actual ERD.

8. References

[1] "Lambda Function Input Event and Response Format ", AWS, 2020. [Online]. Available: <https://docs.aws.amazon.com/lex/latest/dg/lambda-input-response-format.html>. [Accessed: 3- Jul-2020].

[2] "Reading and Writing JSON to a File in Python", AWS, 2020. [Online]. Available: <https://www.geeksforgeeks.org/reading-and-writing-json-to-a-file-in-python/?ref=rp> [Accessed: 12-Oct- 2020].

[3] " Data visualization with different Charts in Python ", AWS, 2020. [Online]. Available: <https://www.geeksforgeeks.org/data-visualization-different-charts-python/>[Accessed: 12- Oct- 2020].

[4] " Generate a graph using Dictionary in Python ", AWS, 2020. [Online]. Available: <https://www.geeksforgeeks.org/generate-graph-using-dictionary-python/>[Accessed: 12- Oct- 2020].

[5] "SQL validator ", AWS, 2020. [Online]. Available: <https://pypi.org/project/sqlvalidator/> [Accessed: 13-Oct- 2020].

[6]" Data dictionary", AWS,2020. [Online]. Available:<https://www.tutorialspoint.com/Data-Dictionary-in-DBMS> [Accessed: 13- Oct- 2020].