Project Final Report

DSCI 551 – Fall 2024

ChatDB: A SQL like chat database software based on console interaction

Group Number: ChatDB 25

Group Member: Liqiang Deng (liqiangd@usc.edu)

Table of Contents

Introduction

This project mainly focuses on developing a kind of interactive ChatGPT-like application that can help user to interact with the database. They can learn how to query data in database system, and perform some simple operations through dialogue with the program. As this is a single person team project, the program currently only supports console terminal interaction and only supports MySQL database. After entering the main program, user can create table structures, upload data from csv files, explore databases and show table, obtain sample queries, execute the queries directly, and ask questions in natural language.

Planned Implementation

Prepare the dataset and design the database system

I choose Python as the programming language, and users can interact through the console. Programming intermediate layer code mobilizes database resources, returns and displays.
As I am a single member team, I have decided to use MySQL as the support. And I will use PyMySQL or mysql-connector-python libraries in the middle layer of the code to interact with the database.
In the proposal plan, it is planned to use three different Kaggle datasets and create a corresponding data table for each dataset. But later on, I made adjustments because I found that I needed to implement SQL statements about 'join', so there needed to be data association between the three data tables. So, I only used the Coffee Sales dataset and manually split the data of the 'Coffee_shop_stales' table into three tables: Stores', 'Products', and 'Transactions' to meet the requirement. Here are the links to the datasets:
https://www.kaggle.com/datasets/ahmedabbas757/coffee-sales
When I deal with the dataset, I may need to perform data cleaning and table partitioning operations, which will be implemented during specific operations. When specific query results need to be returned, generate corresponding SQL statements for querying after matching user input in the middle layer.

Upload data and Explore databases and Show table

User can upload data from local csv files. Users can use 'show table' to view all tables in the database, and input specific number to obtain attribute field information in the chosen tables, as well as obtain some data examples, such as the first 5 items.

Obtain sample queries

Users can enter the example keyword to obtain some SQL query example statements, which are randomly generated according to specific patterns. And these statements can be directly input and executed in this system to obtain corresponding results. The pattern may be group by, having, order by, where, etc. I will support at least 5-10 basic SQL statement keywords.
Users who simply input 'example' related information no patterns will receive several SQL query statements with random patterns.
When the user inputs 'example with [pattern]', several SQL query statements with specific patterns will be randomly generated

Execute the queries

Users can directly input SQL statements for database operations in the console, and only valid SQL statements are supported. Illegal statements will prompt corresponding errors.

Ask questions in natural language

Design several natural language pattern statements in advance, so that when the user inputs the corresponding pattern statements, they can understand the content they want to query, and the query results are returned and displayed. For example, the pattern can be Total <A> by <B>, Count <X> by <Y>, Average <A> by <B>, Max/Min <A> by <B>, Top N <A> by <B>, Filter <A> where <B>. I will implement at least 5 natural language patterns.

Architecture Design

After the user starts running the system's main.py program, a welcome message is displayed. Then users can choose data, currently only MySQL is the only option available. The system basically follows the following process to determine the user's input and return the result. User can exit the program through exit command.

The first figure is the main logic flowchart of the system, which shows the flow logic of the code after the user enters the main program.

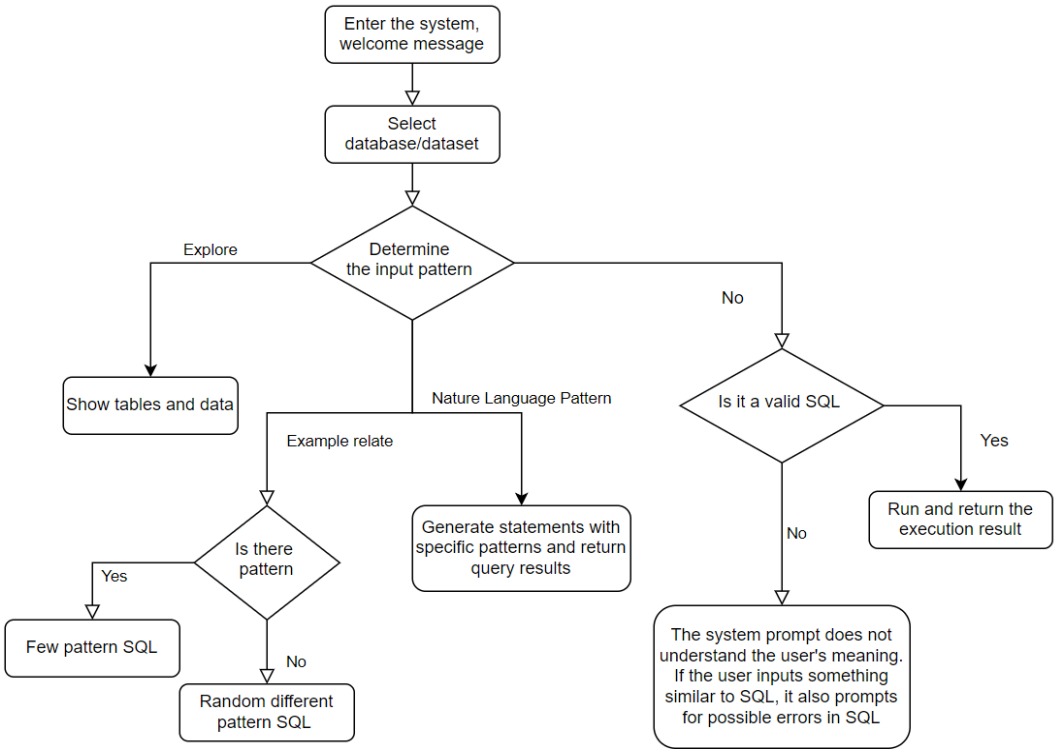
The following is an explanation of the flowchart. The system will enter different method workflows based on user input matching. Firstly, determine whether it is the 'show table' or 'help' command. The former will display the current data table to the user, and based on the user's selection, continue to display the table structure and the first 5 data items of a certain table. The latter will display guidance information to help the user use the system. Afterwards, it will determine whether the SQL queries are valid. If they are, they will be run directly and the result will be returned, including error messages. The next step is to check if the user's input contains the keyword 'example'. If it does, enter the example module to generate SQL query examples and provide explanations for the returned display. If there is no 'example', it will enter the natural language recognition module, attempting to match the user's input pattern and return the SQL queries and explanations that the user wants. Here, the user can choose to run directly and return the results. If none of the above matches are successful and cannot enter any functional module, the user will be prompted that the system cannot recognize the user's intention and hopes that the user can re-enter.

Table Structure:

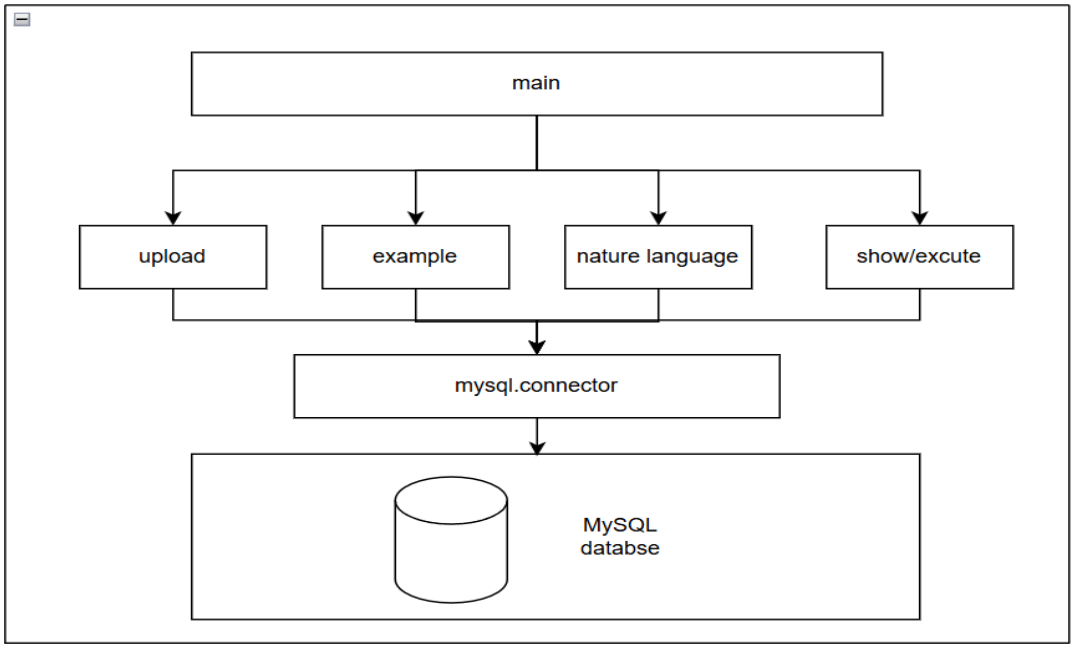
Stores (store_id INT PRIMARY KEY, store_location VARCHAR(255) NOT NULL)

Products (product_id INT PRIMARY KEY, product_detail VARCHAR(255), product_type VARCHAR(255), product_category VARCHAR(255), unit_price DECIMAL(10, 2))

Transactions (transaction_id INT PRIMARY KEY, transaction_date DATE NOT NULL, transaction_time TIME NOT NULL, transaction_qty INT NOT NULL, store_id INT REFERENCES Stores(store_id), product_id INT REFERENCES Products(product_id))



The following is the system architecture diagram. Users interact through the main process, which calls different functional modules. The functionality of each module interacts with the underlying MySQL database through a unified 'mysql.connector' intermediate layer.



Implementation

Functionalities

- Upload data

Users can select CSV files through tkinter's filedialog to upload data. After selecting the file, there will be a simple verification function that checks whether the column names in the CSV file correspond one-to-one with the data table organization, and verifies whether the data type and range meet the requirements. The program will split the data into three tables ('Stores', 'Products', and 'Transactions') and insert them into the database. By the way, users can directly run the corresponding file outside the main process to create the specified table structure into the MySQL database.

- Show tables

Users can view the tables in the currently selected database by entering the 'show table' command. The table names will be returned with numbers, and users can enter the corresponding numbers to display the detailed table structure of the corresponding table and the first 5 pieces of data.

- Help Guidelines

Users can view the supported functions and guidance of the current system by entering the 'help' command.

- Obtain sample queries

Users can tell the system to enter the example module by including the keyword 'example' in their input. At this point, the system will extract whether the user's input contains the keyword 'group by', 'having','order by','when','limit ','avg','sum','join'(currently implemented keyword types). The program will iterate through the included keywords and output 3 SQL queries with explanations for each type. If no keywords are specified, 3 examples will be randomly generated using the keywords. The process of generation involves randomly selecting fields and values within a limited range, so each generated SQL query is different and can run smoothly.

- Directly execute SQL queries

Users can input executable SQL queries to have the system run directly and return the running results. The system will check if it is a valid SQL, and if it is not a valid SQL, an error will be reported to remind the user.

- Obtain queries from nature language process

If there are no matching prompt keywords for other commands and functions, the system will enter the natural language inference module. Then the system will use re to match patterns from user input content. The currently supported modes are Total<A>by<B>, Count <X> by <Y>, Average <A> by <B>, Max/Min <A> by <B>, Top N <A> by <B>, Filter <A> where <B>, Highest/Lowest <A> for <B>. There are 9 types. The program will first match the corresponding patterns, extract the contents of N, A, and B, and map them to the corresponding data tables and fields through dict. Then, it will determine whether a join is needed based on the table names involved, and obtain the conditions for the join connection. Finally, combine all the information to obtain the required SQL queries and corresponding explanations, and return them to the user. Afterwards, users can choose whether to directly run the corresponding SQL queries.

Tech Stack

I have been using Python (Python = 3.10) to program, using mysql-connector-python to connect to the database, currently only MySQL (Ver 8.0.34) is supported, using tkinter for upload GUI interaction, using random library for random generate related value in example SQL queries, using re for matching patterns in SQL example and user nature language parsing, using csv package to handle raw dataset. Ignore some commonly used packages. The technology stack used in the entire project is very simple and conventional.

Implementation Screenshots

Enter the ChatDB

```
(dsci) C:\Users\admin\Desktop\DSCI551\project>python main.py
-----

Welcome to ChatDB!

-----

This is a database that records data related to coffee shop sales.
You can view the table structure of the relevant data tables through terminal interaction,
fill in the data according to the requirements of the CSV file, upload it to the system,
output example SQL, execute relevant legal SQL, and generate SQL query results according to your ideas,
or you can follow the prompts of the system to operate.
You can type 'help' to see guildline. Anyway, have fun!

-----


Please select the database:
1. MySQL

Enter your choice (1 to select MySQL):
```

Show tables

```
Enter your query or chat (or type 'exit' to quit): show table
Showing available tables:

The available tables are as follows:
1. products
2. stores
3. transactions

Please enter the table number to view the table structure, or enter 'b' to return: 1

The Structure of Table products:
------------------------------------------------
Field: product_id, Type: int, Is NULL: NO, Key Type: PRI
Field: product_detail, Type: varchar(255), Is NULL: YES, Key Type:
Field: product_type, Type: varchar(255), Is NULL: YES, Key Type:
Field: product_category, Type: varchar(255), Is NULL: YES, Key Type:
Field: unit_price, Type: decimal(10,2), Is NULL: YES, Key Type:
------------------------------------------------

Sample Data from Table products:
------------------------------------------------
product_id      product_detail  product_type    product_category        unit_price
------------------------------------------------
1       Brazilian - Organic     Organic Beans   Coffee beans    18.00
2       Our Old Time Diner Blend        House blend Beans       Coffee beans    18.00
3       Espresso Roast  Espresso Beans  Coffee beans    14.75
4       Primo Espresso Roast    Espresso Beans  Coffee beans    20.45
5       Columbian Medium Roast  Gourmet Beans   Coffee beans    15.00
------------------------------------------------
```

Upload data validation failed situation

```
Enter your query or chat (or type 'exit' to quit): upload
Please select a CSV file...
Unexpected error: Field name validation failed. Missing fields: transaction_id; Unexpected fields: transaction_ids
Please try selecting another file.
```

Upload data validation success situation

```
Enter your query or chat (or type 'exit' to quit): upload
Please select a CSV file...
Field names validation passed.
Data inserted successfully.
```

Obtain sample queries

```
Enter your query or chat (or type 'exit' to quit): example

Here are some example SQL queries with explanations:

Query: SELECT product_type, SUM(unit_price) AS total_unit_price FROM products GROUP BY product_type;
Explanation: This query groups the data by product_type and calculates the sum of unit_price for each product_type.

Query: SELECT * FROM transactions LIMIT 10;
Explanation: This query retrieves the first 10 records from the transactions table.

Query: SELECT * FROM stores LIMIT 2;
Explanation: This query retrieves the first 2 records from the stores table.
```

Obtain sample queries with specific language constructs

```
Enter your query or chat (or type 'exit' to quit): example group by join

Here are some example SQL queries with explanations:

Query: SELECT product_category, COUNT(unit_price) FROM products GROUP BY product_category;
Explanation: This query groups the data by product_category and calculates the count of unit_price for each product_cate
gory.

Query: SELECT product_category, SUM(unit_price) FROM products GROUP BY product_category;
Explanation: This query groups the data by product_category and calculates the sum of unit_price for each product_catego
ry.

Query: SELECT product_type, AVG(unit_price) FROM products GROUP BY product_type;
Explanation: This query groups the data by product_type and calculates the avg of unit_price for each product_type.

Query: SELECT transactions.store_id, stores.store_id FROM transactions JOIN stores ON transactions.store_id = stores.sto
re_id;
Explanation: This query retrieves data by joining 'transactions' with 'stores' on the condition 'transactions.store_id =
 stores.store_id'. It selects 'store_id' from 'transactions' and 'store_id' from 'stores'.

Query: SELECT transactions.transaction_date, stores.store_location FROM transactions JOIN stores ON transactions.store_i
d = stores.store_id;
Explanation: This query retrieves data by joining 'transactions' with 'stores' on the condition 'transactions.store_id =
 stores.store_id'. It selects 'transaction_date' from 'transactions' and 'store_location' from 'stores'.

Query: SELECT transactions.transaction_id, stores.store_id FROM transactions JOIN stores ON transactions.store_id = stor
es.store_id;
Explanation: This query retrieves data by joining 'transactions' with 'stores' on the condition 'transactions.store_id =
 stores.store_id'. It selects 'transaction_id' from 'transactions' and 'store_id' from 'stores'.
```

Ask questions in natural language (total A by B)

```
Enter your query or chat (or type 'exit' to quit): find total unit price by product
Generated SQL: SELECT product_detail, SUM(unit_price) AS total FROM products GROUP BY product_detail;
Explanation: This query groups the data by product and calculates the total unit price for each product.

You can directely input valid SQL to retrieval related information.
Let me know if you'd like further assistance or a different analysis!

Would you like to directly execute this SQL? (yes/else to skip): y
('Brazilian - Organic', Decimal('18.00'))
('Our Old Time Diner Blend', Decimal('18.00'))
('Espresso Roast', Decimal('14.75'))
('Primo Espresso Roast', Decimal('20.45'))
('Columbian Medium Roast', Decimal('15.00'))
('Ethiopia', Decimal('21.00'))
('Jamacian Coffee River', Decimal('19.75'))
```

Ask questions in natural language (highest A for B, a more complex scenario involving joins)

```
Enter your query or chat (or type 'exit' to quit): highest sales day for store
Generated SQL: SELECT stores.store_location, transactions.transaction_date, SUM(transactions.transaction_qty * products.
unit_price) AS total FROM transactions JOIN stores ON stores.store_id = transactions.store_id JOIN products ON products.
product_id = transactions.product_id GROUP BY transactions.transaction_date, stores.store_location ORDER BY total DESC;
Explanation: This query groups the data by store and calculates the highest sales day for each store, using a JOIN conne
ct transactions, stores and products tables.

You can directely input valid SQL to retrieval related information.
Let me know if you'd like further assistance or a different analysis!

Would you like to directly execute this SQL? (yes/else to skip): y
("Hell's Kitchen", datetime.date(2023, 2, 19), Decimal('149.75'))
('Astoria', datetime.date(2023, 3, 15), Decimal('137.20'))
```

Learning Outcomes

Challenges Faced

The difficulties involved mainly come from two aspects. The first aspect is that we are not allowed to use NLP language models to infer user input. The language input by users involves different languages and various expressions, and the forms of words are also different, making it difficult to accurately obtain the user's intention and corresponding target objects. So, I only used a relatively clumsy method, enumerating a large number of patterns that users may use and the possible different forms of expressions for corresponding fields, and mapping them all to the corresponding data tables and fields to make the system understand. The second aspect comes from possible join operations, where the user's intention may involve data from multiple tables to participate in completing the search. There will be a maximum of 3 tables in this system. I obtained the join conditions by first extracting the involved fields and data tables, and then designed a separate SQL template for the join to complete this task.

Individual Contribution

As this is a single person team, I am responsible for all the work of the project. Including system design, data collection, data cleaning, code writing, database system implementation, language pattern matching, exception handling, project document writing, etc.

Conclusion

Through this ChatDB project, I completed the development of an interactive database program and obtained a system that can help users better use the database and teach them how to generate SQL queries. Specifically, with the continuous development of data science. Data will become increasingly important in our production and daily life, and there will be more people from different background who need to understand how to use databases. Developing such a system is very useful. Through the conceptualization and brainstorming of this project, I gained a deep understanding of how to design an intermediate program to interact with the current MySQL database, as well as how to design a system that is more user-friendly. Although the system did not use advanced technology and implemented basic functions with some simple technology stacks, it still feels great to be able to complete a complete project.

Future Scope

During my development process, due to time constraints, I had many ideas that I didn't have the chance to implement. These were the optimizable points I discovered outside of the proposal. Firstly, we can add more patterns to match users' natural language input, or in the future, we can introduce NLP language models to improve this project. Secondly, we should be able to support more filtering conditions, such as'>','>=','<>','<','<=', but now we only support the case of'='. Finally, we may be able to support users to modify in the future, delete data, and modify table structures. After improving the above points in the future, adding UI interfaces and supporting more other databases, I believe the system will have comprehensive functions for interacting

with databases, and similar applications will become increasingly popular in the future.

Appendix

GitHub Link: https://github.com/LiqiangDeng/chatdb_sql_terminal

Presentation Slide Link: https://docs.google.com/presentation/d/1rfKO-uPIyY3sdlhUigY_3p9Zeu8UdhalLWrYnJ0aCW4/edit#slide=id.p