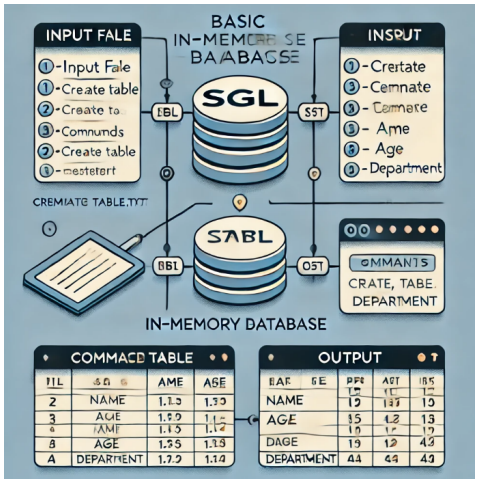
PROJECT ASSIGNMENT 3

Issue Date : 22.11.2024
Recitation Date : 22.11.2024 - Friday (15:00) (held on Zoom)
Due Date : 08.12.2024 (23:00:00)
TAs : R.A Alpay TEKİN
Programming Language : Python 3.9.18



1 Introduction

A database is an organized collection of data stored in a computer system and usually controlled by a database management system (DBMS). The data in common databases is modeled in tables, making querying and processing efficient. Structured query language (SQL) is commonly used for data querying and writing[1].

This assignment introduces the concept of database management through simulation. You will build a basic in-memory database system that supports fundamental operations such as CREATE, INSERT, DELETE, SELECT, UPDATE, COUNT, and JOIN. The purpose is to help you understand the concepts of data manipulation and querying programmatically without relying on an external database management system.

Each of these operations will need to be implemented as Python functions. For example, the CREATE operation should define the structure of a table, INSERT should allow adding records, DELETE should remove specific entries, and SELECT should retrieve data based on given conditions. Similarly, functions for UPDATE, COUNT, and JOIN will enable modifying records, counting entries, and combining data from multiple tables, respectively. Through this assignment, you will gain hands-on experience in simulating the core functionalities of a database management system.

2 Objectives

You must implement a Python program that:

- Manages multiple tables with column definitions and data.
- Supports inserting, deleting, updating, and querying rows from these tables.
- Implements filtering using conditions for all supported operations. **Conditions are limited to equality comparison. You do not need to implement other SQL filtering operations(i.e., not equal, lower than)**
- Performs table joins on a specified column (on one column only).
- Produces output in a structured, readable format.
- Handles errors gracefully, such as:,
  - Attempting to access a non-existent table.
  - Querying or updating with a non-existent column.
  - Providing incorrect syntax or invalid input for commands.
- Exceptions will be written for these errors, and **try-catch blocks** will be used.
- The above-mentioned operations must be implemented as separate Python functions, with each function dedicated to handling its specific operation in the database system.

3 Specifications

The objective of this project is to execute SQL-like commands that are provided in a given input file. Your program must read and process these commands sequentially, execute the specified operations on an in-memory database, and generate appropriate outputs. Each command should be executed independently, and the results should be displayed in a clear and structured format.

3.1 Supported Commands

3.1.1 CREATE TABLE

Define a table with a list of column names.

```
CREATE_TABLE <table_name> <columns>
CREATE_TABLE employees name,age,department
```

3.1.2 INSERT

Add a row of values to a specified table.

```
INSERT <table_name> <data>
INSERT employees John,30,Engineering
```

3.1.3 SELECT

Retrieve rows based on column names and/or conditions. If no columns are specified, select all columns.

```
SELECT <table_name> <columns> WHERE <conditions>
SELECT employees name,age WHERE {"deparment": "Engineering"}

// multiple conditions
SELECT employees name,age WHERE {"deparment":"Engineering", "age":"22"}

// select all columns
SELECT employees * WHERE {"deparment": "Engineering"}
```

3.1.4 UPDATE

Modify rows in a table based on conditions.

```
UPDATE <table_name> <updates> WHERE <conditions>
UPDATE employess {"age": "13"} WHERE {"deparment": "Engineering"}

// update multiple fields
UPDATE employess {"age": "45", "name": "Alan"} WHERE {"age": "30"}
```

3.1.5 DELETE

Remove rows matching given conditions. If no condition is provided, all rows will be deleted.

```
DELETE <table_name> WHERE <conditions>
DELETE students WHERE {"age": 22}

// multiple conditions
DELETE employess WHERE {"age": "30", "name": "Alan"}
```

3.1.6 JOIN

Combine rows from two tables based on a common column.

```
JOIN <table1>,<table2> ON <column>
JOIN students,courses ON major
```

3.1.7 COUNT

Return the count of rows satisfying a given condition.

```
COUNT <table_name> WHERE <onditions>
COUNT students WHERE {"age": 22}

// multiple conditions
COUNT employess WHERE {"age": "30", "name": "Alan"}
```

3.2 Input and Output

3.2.1 Input

The input file contains a series of SQL-like commands, each on a separate line. Your program must process these commands sequentially. You can find content of sample input text file below:

```
CREATE_TABLE students id,name,age,major
INSERT students 1,John Doe,20,CS
INSERT students 2,Jane Smith,22,EE
SELECT students id,name WHERE {"major": "CS"}
UPDATE students {"major": "SE"} WHERE {"name": "John Doe"}
DELETE students WHERE {"age": 22}
COUNT students WHERE {"major": "CS"}
```

3.2.2 Output

You need to print the tables and other essentials in format that is shown in sample output.

```
###################### CREATE ##########################
Table 'students' created with columns: ['id', 'name', 'age', 'major']
########################################################

###################### INSERT ##########################
Inserted into 'students': ('1', 'John Doe', '20', 'CS')

Table: students
+----+----------+-----+-------+
| id | name     | age | major |
+----+----------+-----+-------+
| 1  | John Doe | 20  | CS    |
+----+----------+-----+-------+
########################################################

###################### INSERT ##########################
Inserted into 'students': ('2', 'Jane Smith', '22', 'EE')

Table: students
+----+------------+-----+-------+
| id | name       | age | major |
+----+------------+-----+-------+
| 1  | John Doe   | 20  | CS    |
| 2  | Jane Smith | 22  | EE    |
+----+------------+-----+-------+
########################################################
```

```
###################### SELECT ########################
Condition: {'major': 'CS'}
Select result from 'students': [('1', 'John Doe')]
######################################################

###################### UPDATE ########################
Updated 'students' with {'major': 'SE'} where {'name': 'John Doe'}
1 rows updated.

Table: students
+----+------------+-----+-------+
| id | name       | age | major |
+----+------------+-----+-------+
| 1  | John Doe   | 20  | SE    |
| 2  | Jane Smith | 22  | EE    |
+----+------------+-----+-------+
######################################################

###################### DELETE ########################
Deleted from 'students' where {'age': 22}
1 rows deleted.

Table: students
+----+----------+-----+-------+
| id | name     | age | major |
+----+----------+-----+-------+
| 1  | John Doe | 20  | SE    |
+----+----------+-----+-------+
######################################################

###################### COUNT #########################
Count: 0
Total number of entries in 'students' is 0
######################################################
```

4 Execution and Test

Your code must be executed under **Python 3.9.18 at dev.cs.hacettepe.edu.tr**. If your code does not run at department's developer server during the testing stage, then you will be graded as 0 for the code part even if it works on your own machine. Sample command as follows:

```
python3 database.py input.txt
```

5 Grading

Task	Grade
CREATE TABLE (Part 3.1.1)	5
INSERT (Part 3.1.2)	5
SELECT (Part 3.1.3)	10
UPDATE (Part 3.1.4)	10
DELETE (Part 3.1.5)	10
COUNT (Part 3.1.6)	10
JOIN (Part 3.1.7)	10
Error Handling and Exceptions	20
Clean Code and Comments	20*
Total	100

*** The score of the clean code comment part will be multiplied by your overall score (excluding clean code comment part) and divided by the maximum score that can be taken from these parts. Say that you got 60 from all parts excluding clean code comment part and 10 from clean code comment part, your score for clean code comment part is going to be 10*(60/80) which is 7.5 and your overall score will be 60+7.5=67.5.**

Note that you must score one at the submit system, otherwise 20% of your grade will be deducted, moreover, usage of global variables are forbidden and you must implement a main function as advised otherwise 20% of your grade will be deducted! There may also be other point deductions if you do not obey the given rules, such as if you do not use functions and/or loops as necessary.

6 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only not compressed as .zip files because the system just supports .zip files).

```
-b<studentID>.zip
  -database.py
```

7 Late Policy

You have two days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You must submit your solution in at the most two days later than submission date, otherwise it will not be evaluated. Please do not e-mail to me even if you miss the deadline for a few seconds due to your own fault as it would be unfair for your friends, e-mail submissions will not be considered if you do not have a valid issue.

8 Notes and Restrictions

- Your code must be able to execute on our department's developer server (dev.cs.hacettepe.edu.tr).
- You must use the primitive Python data structures that you've learned in lectures(array, list, dictionary, tuples).
- You must use comments for this project and you must give brief information about the challenging parts of your code. Do not over comment as it is against clean code approach. Design your comments so that they make your code fully understandable and not excessive for others. You can check guides of Python namely PEP-8 and PEP-257 for further information.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as "clean code".
- Use UNDERSTANDABLE names to your variables, classes, and functions regardless of the length. The names of functions, attributes and classes should obey Python naming convention. This expectation will be graded as "coding standards".
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- Do not miss the submission deadline.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- You can ask your questions through course's Piazza group, and you are supposed to be aware of everything discussed in the Piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.

References

[1] *What is Database?* Accessed November 18, 2024. 2024. URL: `https://www.geeksforgeeks.org/what-is-database/`.